An Optimization Tool for Local Customized Object Detector in Edge Devices

Jang Woon Baek, Yun Won Choi, Jinhong Kim, Joon-Goo Lee
Daegu-Gyeongbuk Research Center
Electronics and Telecommunications Research Institute
Daegu, Korea
{jwbaek98, yunwon.choi, jinhong, leejg01679}@etri.re.kr

***Abstract*—In this paper, we propose an optimization tool for local customized object detector which has a light weight deep learning model and operates on the inexpensive and low-performance edge devices. We focused on the edge devices which provide video analysis services on a CCTV camera that monitors a specific area. The pre-trained object detection model needs customizing processes such as re-learning depending on the camera location in order to reduce false alarm and miss detection. That requires additional time and energy to create a training DB from the video of the CCTV camera and to train the detection model using the created DB. In order to reduce this effort, we developed a web-based optimization tool for a local customized object detector. The proposed tool provides the automatic DB generation and re-learning functions with user friendly interfaces. We can see that the proposed tool is very simple and efficient which only requires a video file from the local camera and the pre-trained detection model, while automatic DB generation and re-learning processes are done internally.**

Keywords—Local customization; DB generation tool; light weight detector

I. Introduction

The demand for intelligent video analysis services on edge devices is increasing. Edge devices need an accurate and lightweight object detector to provide video analysis services on low-cost, low-performance devices[1]. Lightweight object detectors have poor detection accuracy depending on camera specifications, installation location, and weather conditions. In general, to solve this problem, an optimization process is performed by creating a learning DB from CCTV camera images and re-training the object detector using the created DB. Figure 1 shows the comparison of object detection results before and after object detector optimization. The right side of Figure 1 shows the detection result of the object detector using the pre-trained model based on Open DB. There are many false detections and miss detections when object detection is performed with a pre-trained model. However, when the object detector is optimized with CCTV camera images, false detection and miss-detection are reduced as shown on the right side of Figure 1.

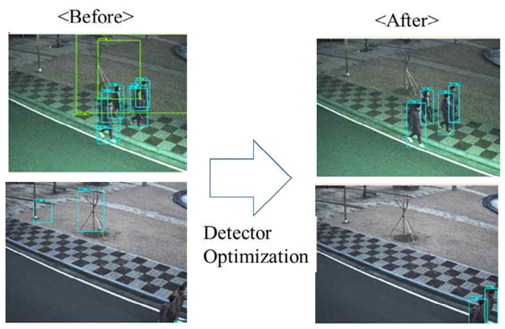


Fig. 1. An example of the result of optimizing the object detector

We focused on an edge device that receives videos from the CCTV camera, which monitors a fixed area, and provides intelligent video analysis services. Since the edge device only needs to detect objects on the fixed monitored area, there is no need for a detection model with high generalization performance that detects well in surveillance areas of almost all of the cameras. Even if a lightweight model is used in the edge devices, it is possible to provide a high-accuracy object detection for the fixed monitored area by optimizing the detector with the images from the local camera. However, in order to optimize the object detector of edge devices, it takes additional time and energy to create a training DB and re-learn a detection model for each CCTV camera. In particular, it is very cumbersome and difficult for a person to create a learning DB from CCTV camera images.

In order to solve these problems, we propose a tool for optimizing a local customized object detector. The proposed tool automatically creates a learning DB using video streams acquired from a camera installed in the field, and optimizes the local customized object detector by re-training the object detector based on the created DB. The our previous research proposed an automatic database generation algorithm, which creates a learning DB by extracting the background from the field image and synthesizing the extracted background and the detection results of the pre-trained object detector[2]. The proposed tool uses the previous researched algorithm for DB generation. In our proposed tool, a high-quality learning DB can be created by adjusting the object detection threshold. The created training DB includes a composite of an image and a annotation file for the target object. Then, the object detector is re-trained based on the created learning DB. At this time, the object detector is learned by extracting the feature map from the existing pre-learning model and using it as an initial value, and the generated DB path is used as an input. After re-training, the detection result image before/after object detector optimization can be compared by the tool. The tool can be installed and used on a local personal PC, or can be used by several people at the same time remotely in the form of a web service by accessing a server.

II. Optimization Tool

Figure 2 shows the configuration of the optimization tool for a local customized object detector. The optimization tool consists of a DB generation module and a training module which learns an object detector using the generated DB. For DB generation, the optimization tool receives the video images of a CCTV camera and the pre-trained model as input file. The DB generation module automatically creates a DB consisting of a pair of an image and a tagging file. The training module relearns the object detector using the automatically generated DB. The DB generation and training processes can be repeated several times depending on the degree of optimization of the object detector. Finally, we can obtain the optimized detection model for the local surveillance area.

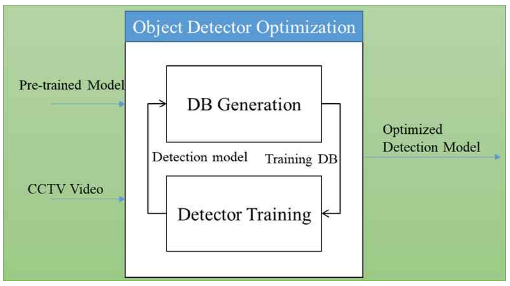


Fig. 2. The configuration of the optimization tool for a local customized object detector

Figure 3 shows the processes for DB generation. The DB generation module consists of the three processes which are a background modeling process, an object detection process, and a synthesizing process. The background modeling process extracts the background images from the input video images. The object detection process detects the target objects on image frame from CCTV video with a pre-trained detection model. The synthesizing process combines the background image and detected object regions, and generates the annotation files. The input video images can be video files (*.avi, .mp4, etc*) or video streaming URL address (*rtsp://xxx.xxx.xxx*).

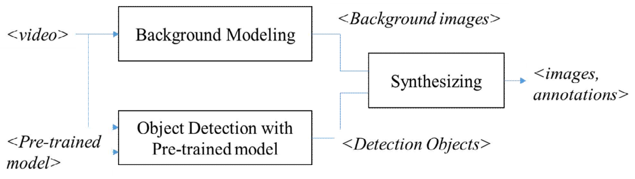
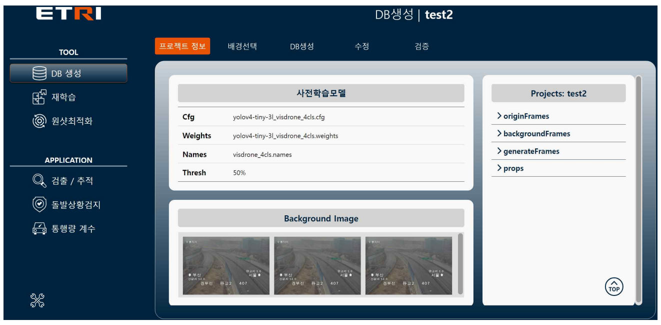


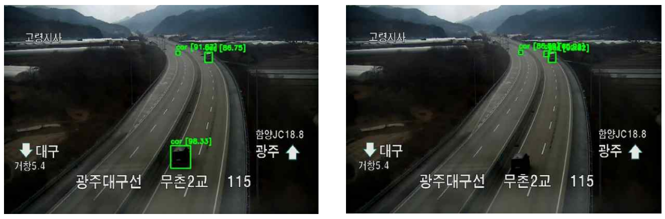
Fig. 3. Automatic DB generation processes

Automatic DB generation processes create an image by synthesizing objects detected by the pre-learning model in the background, and create annotation files (object class, object area) for target objects. At this time, the class of detection object to be reflected in DB generation can be selected. For example, in the case of a 4-class (person, car, bicycle, motorcycle) detector, only 2-class (person, car) of detection results can be reflected in DB generation. Therefore, it is possible to create training DBs for only wanted target objects. In addition, the object detector with the pre-trained model may have false detections in the background image. In this case, it should be removed in the tagging file, so the tool can set an ignore region where detection results are not reflected in DB creation. The case where the ignore area and IoU (intersection over union) is over a certain threshold is excluded. Also, the proposed tool can collectively remove falsely detected objects in the background part from the entire tagging files.

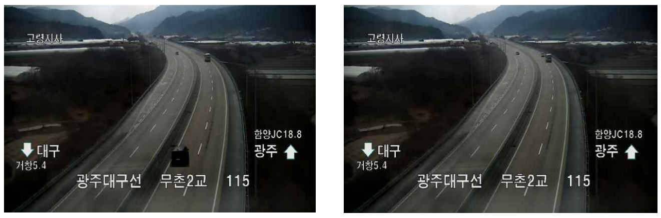


Cfg	yolov4-tiny-3l_visdrone_4cls.cfg
Weights	yolov4-tiny-3l_visdrone_4cls.weights
Names	visdrone_4cls.names
Thresh	50%

(a) Background modeling and selection



(b) Object Detection with Pre-trained Model

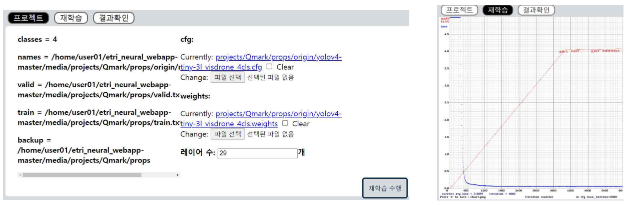


(c) Synthesizing a background and target objects

Fig. 4. An example of the results of automatic DB generationp processes

Figure 4 shows an example of the results of automatic DB generation processes. Our tool creates a project for the optimization of the local customized object detector. When a project is created, a pre-trained detector model and a video file from the target camera are loaded. And then, background modeling is performed, and we can select the background image from the results of the background modeling. The DB generation are performed automatically with synthesizing the background image and the objects detected by pre-trained model. At this time, the tagging files, which contain the class and location information of the object, are generated simultaneously. The tagging files are generated with *yolo* annotation format [3,4]. In this example, we used the yolov4-tiny [5] model that is pre-trained using COCO[6] and visdrone database[7].

When the automatic DB generation is completed, re-learning processes proceed. Figure 5 shows the configuration for the re-learning processes. The generated DB paths are used as inputs for training and validation. And the partial pre-trained model can be used for the re-learning as the initial value for optimization. The re-learning is very fast because the light weight network model is used, and the target area DB is much smaller than the open DB such as COCO. The tool provides a graph image of learning progress (Loss, mAP). When learning is completed, an optimized object detector model is created.



classes = 4

names = /home/user01/etri_neural_webapp-master/media/projects/Qmark/props/origin/r

valid = /home/user01/etri_neural_webapp-master/media/projects/Qmark/props/valid.tx

train = /home/user01/etri_neural_webapp-master/media/projects/Qmark/props/train.tx

backup = /home/user01/etri_neural_webapp-master/media/projects/Qmark/props

cfg:

Currently: projects/Qmark/props/origin/yolov4-tiny-3l_visdrone_4cls.cfg ☐ Clear

Change: 파일 선택 선택된 파일 없음

weights:

Currently: projects/Qmark/props/origin/yolov4-tiny-3l_visdrone_4cls.weights ☐ Clear

Change: 파일 선택 선택된 파일 없음

레이어 수: 29 개

Fig. 5. Re-learning Processe

Using our tool, we can easily confirm the result of the optimized object detection model. Also, we can compare and display detection results before/after optimization for field images. Figure 6 shows an example of before and after of the optimization of the local customized object detector. We can see that false alarms and miss detections are reduced after the optimization. Additionally, we installed edge cameras in an actual test area and tested field-customized object detector optimization technology (Fig. 7). It was applied to vehicle speed detection and unexpected situation detection applications. It was confirmed that application performance improved due to improved object detection rate.

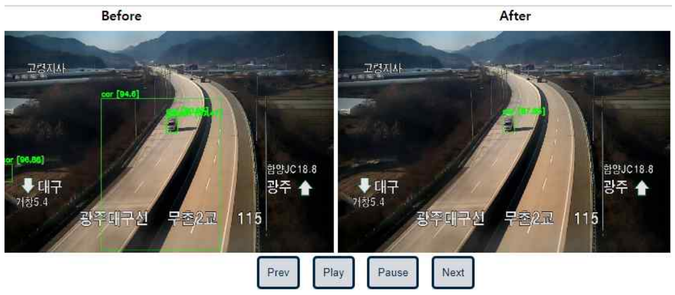


Fig. 6. Berfore and after of the optimization of the local customized object dectector

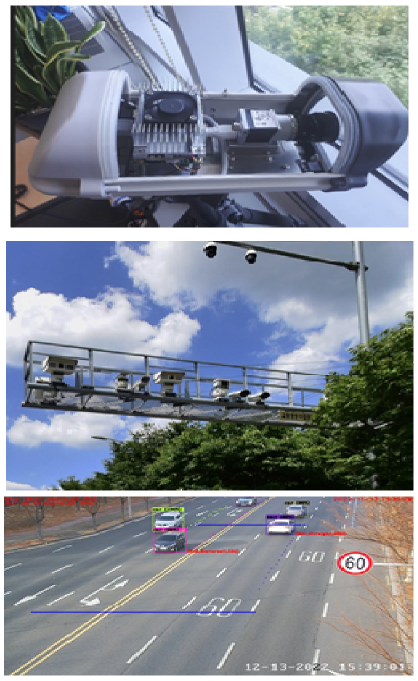


Fig. 7. Real test environment for local customized object detector of vehicle speed and abnormal events detection applications on edge devices

ACKNOWLEDGMENT

This work was supported by the Commercialization Promotion Agency for R&D Outcomes(COMPA) grant funded by the Korean Government(Ministry of Science and ICT). (RS-2023-RS-2023-00304776)

REFERENCES

[1] Chan Yung Kim, Kwi Seob Um, Seo Weon Heo “A novel MobileNet with selective depth multiplier to compromise complexity and accuracy,” ETRI Journal, Oct. 2022.

[2] Lee, J. G., and Baek, J. W. "An Automatic Database Generation Algorithm for Local Optimization of CNN Object Detector for Edge Devices," 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Nov. 2020.

[3] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," arXiv:1506.02640 [cs.CV]

[4] Joseph Redmon, Ali Farhadi, "YOLO9000: Better, Faster, Stronger," arXiv:1612.08242 [cs.CV]

[5] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934 [cs.CV]

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, Piotr Dollár, "Microsoft COCO: Common Objects in Context," arXiv:1405.0312 [cs.CV]

[7] Zhu, Pengfei and Wen, Longyin and Du, Dawei and Bian, Xiao and Fan, Heng and Hu, Qinghua and Ling, Haibin, “Detection and Tracking Meet Drones Challenge,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021