Efficient Residual Bottleneck for Object Detection on CPU

Jinsu An, Muhamad Dwisnanto Putro, Kang-Hyun Jo
Department of Electrical, Electronic and Computer Engineering, University of Ulsan
Ulsan, South Korea
jinsu5023@islab.ulsan.ac.kr, dwisnanto.putro@gmail.com, acejo@ulsan.ac.kr

***Abstract*—Object detection is the most fundamental and important task in computer vision. With the development of hardware such as computing power of GPUs and cameras, object detection technology is gradually improving. However, there are many difficulties in using GPUs in industrial fields. Therefore, it is very important to use efficient deep learning technology in the CPU environment. In this paper, we propose a deep learning model that can detect objects in real-time from images and videos using CPU. By modifying the CSP [1] bottleneck, which corresponds to the backbone of YOLOv5 [2], an experiment was conducted to reduce the amount of computation and improve the FPS. The model was trained using the MS COCO dataset, and compared with the original YOLOv5, the number of parameters was reduced by about 2.4%, and compared with RefineDetLite, the mAP value was measured to be 0.367 mAP, which is 0.071 higher than that of RefineDetLite. The FPS was 23.010, which was sufficient for real-time object detection.**

***Index Terms*—Object Detection, Real-time Detection, CPU, Low-cost, Efficiency**

I. Introduction

Object detection is the most fundamental and important task in computer vision to discriminate the position and class of an object in an image. It has been continuously studied over the past few years and essential research field that can be applied in various fields such as surveillance systems, autonomous driving, and robotics, and is the basis of other computer vision studies. The rapid development of deep learning technology over the past few years has greatly improved object detection technology. Due to the deep learning technology and the computing power of GPUs, the performance of object detection has been greatly improved, resulting in significant results in the field of computer vision. Many applications, such as medical monitoring, autonomous driving, intelligent surveillance systems, anomaly detection, and robotic vision, are built on deep learning object detection.

Advances in hardware such as (Graphic Processing Unit) GPU computing power and cameras have had a significant impact on the improvement of object detection technology. Cameras are getting smaller and cheaper, higher resolution, and GPU computing power is getting bigger and more efficient. These advances in hardware have made it possible to perform computer vision with real-time object detection and tracking. Many object detection methods utilize deep learning technology and GPU computing power to produce fast and good results, but these methods are too difficult to use in industrial fields. This is because it is too expensive to utilize the computing power of the GPU in the industrial field. Using deep learning technology in a CPU environment is much slower than using the computing power of the GPU and the results are not good. It creates dependence on the expensive devices. Research to make a deep learning network lighter, faster, and more efficient using efficient deep learning technology in a CPU environment is very important.

The object detection task has developed over the past 20 years and is generally divided into two methods. It is a method of traditional image processing and deep learning. There are two types of deep learning methods, one-stage and two-stage methods. The method proposed in this paper is a one-stage-based deep learning method. One-stage-based object detection predicts the bounding box of an image without a region suggestion step. Therefore, it can be used in real-time applications due to its short calculation time. However, since the operation speed is prioritized, it is difficult to detect irregularly shaped objects or small objects. The advantage of the one-stage method is that it is faster because the structure is relatively simple than the two-stage method.

YOLOv5 has been introduced as excellent object detection, which applies CSP bottleneck to extract the essential features and Path Aggregate Network (PAN) to fuse the information of medium and high-level features. A three convolutional (C3) is used to discriminate the object features against the background efficiently. The C3 layer used in the existing YOLOv5 is a CSP bottleneck with three convolution and consists of a bottleneck and three convolution layers. In order to run the object detection algorithm on the CPU in real-time, it is necessary to reduce the number of parameters in the deep learning object detection network. In this work, we adjusted the convolution of the C3 layer by reducing the number of layer, and replace the order of the concatenation and addition operations of the feature map. The main contributions of this work are as follows:

1. A real-time object detection is proposed to localize the specific object that can be operated on a CPU device.

2. A new structure of the convolutional block is introduced by modifying the fusion operation on the CSP bottleneck module.

Identify applicable funding agency here. If none, delete this.

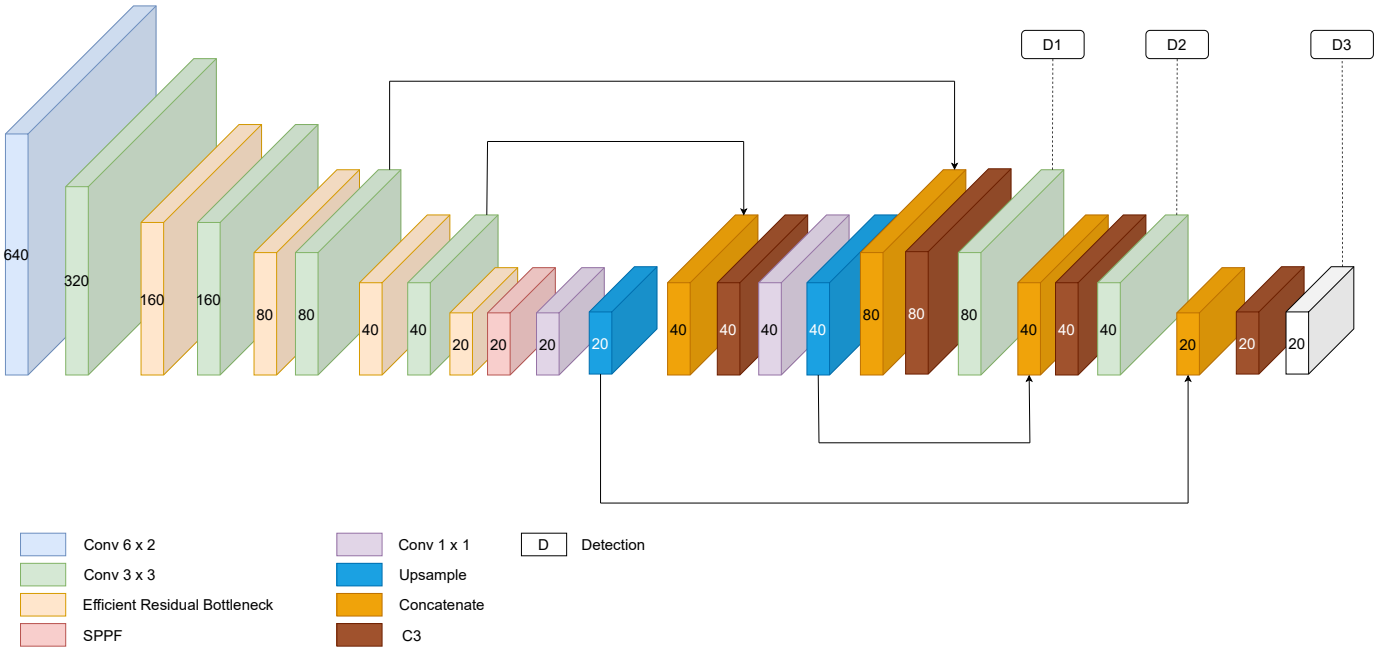

Fig. 1. Proposed Architecture.

II. Proposed Architecture

A. General Architecture

The framework of YOLOv5 has three main components. It is composed of Backbone, Head, and Neck, and Backbone extracts image features and delivers it to the Head through the Neck. Neck creates a feature pyramid by collecting feature maps extracted from Backbone. Finally, it consists of an output layer that detects objects in the head. Among them, the C3 layer used in the backbone was modified to lighten the deep learning object detection model.

B. Efficient Residual bottleneck

The C3 layer is a CSP bottleneck with three convolutions and extracts a total of four features from the backbone in the framework of YOLOv5. To make the C3 layer for extracting features light, we changed the three convolutions to two convolutions, and changed the order of the concatenation and addition operations to match the extracted feature maps.

C. Path Aggregate Network - Multi Detector

Path aggregate network (PANet) [3] is a network used in YOLOv5's Neck. In general, low-level features are useful when detecting small objects, but even when detecting large objects, low-level features that respond strongly to edges or small instances are required. We also need a high-level feature that captures the context of the image to detect small objects. For this purpose, if low-level and high-level features are effectively utilized, more accurate localization is possible. It fuses the feature information from different frequency levels to enrich the knowledge and increase selection performance at the head block. In the existing Feature Pyramid Network (FPN) [4], low-level information is transmitted to high-level through more than 100 layers. In PANet, only about 10 layers are used to transmit low-level information to high-level. This means that low-level features can be transferred to high-level features intact and low-level information can be utilized even when detecting large objects.

D. Multi-box Loss

It uses three loss functions, including IoU loss, binary cross entropy, and confidence loss. The supervision method is adopted from YOLOv5 work that uses Bounding-box regression is the most widely used method in object detection algorithms. It is used to predict the position of an object to be detected using a rectangular box. This method aims to correct the position of the predicted bounding box. Bounding-box regression uses an overlapping area of a ground-truth box and predicted box location, called Intersection over Union (IOU). First, the IoU loss evaluates the difference between the predicted box position and the intersection of the box on the real object, the center point distance, and the aspect ratio. Second, we apply a confidence loss to evaluate whether an object is in each cell. Finally, we measure the error in the probability of the predicted object class using binary cross entropy. Binary cross entropy is applied to effectively measure the probability class of training models, which solve many classification problems simultaneously. The multi-box loss [5] is expressed as follows by combining the above three loss functions.

$$L_{MB} = \lambda_{coord} \sum_{g=1}^{G^2} \sum_{a=1}^{A} \mathbf{g}_{ga}^{obj} L_{coord} + \lambda_{obj} \sum_{g=1}^{G^2} \sum_{a=1}^{A} \mathbf{1}_{ga}^{obj} L_{obj} + \lambda_{cls} \sum_{g=1}^{G^2} \sum_{a=1}^{A} \mathbf{1}_{ga}^{obj} L_{cls}, \tag{1}$$

III. Implement detail

In this session, experiments with the MS COCO dataset are described through the proposed architecture. As an experimental environment, the model was implemented using PyTorch library in a Linux environment. The experiment was performed on Intel Xeon Gold CPU, and Nvidia Tesla V100 32GB GPU was used to train the deep learning model. It uses a Intel I5 6600 @3.3GHz, 32GB RAM as the main CPU to test the speed of the model.

IV. Experimental results

A. Evaluate on dataset

The proposed method tested the object detection performance on the MS COCO2017 dataset. There are a total of 80 different classes in COCO datset, and it consists of a total of 143,575 image data. The COCO dataset contains objects of various sizes, complex backgrounds, and many obstacles. We train the proposed model with 118,287 image data, val the model with 5,000 images, and test the model with 20,288 images. The object detection model extracts and learns the features of various objects included in the dataset, and is evaluated through the dataset. To evaluate the model, we used Average Precision (AP), which measures the accuracy of the predicted bounding box. We derive AP for a total of 80 classes, and finally measure the mean Average Precision (mAP) value for all classes. As a result, the mAP value of the proposed method was measured to be 0.367.

B. Runtime efficiency on CPU

The proposed method focuses on running the real-time object detection algorithm on the CPU. In order for the algorithm to work in real time, the number of parameters must be reduced and the FPS must be high. By modifying the convolution layer of the CSP bottleneck module, it was possible to reduce the number of parameters and improve the FPS. Compared to RefineDetLite, the mAP value was about 0.07 higher, and the FPS at 320x320 resolution was about 15 FPS higher. The number of parameters is also about 2.4% less compared to YOLOv5, so it is possible to obtain an FPS value that can perform object detection in real time while increasing the computational efficiency.

V. Conclusion

In this paper, we propose Efficient Residual Bottleneck for a deep learning algorithm capable of real-time object detection. In order to reduce the amount of computation, the existing CSP bottleneck was corrected, and training was performed on the MS COCO dataset. The mAP value was measured to be 0.367, which was 0.071 mAP higher than RefinedeLite, which was 0.296, and the FPS was 23.010, which was 15.38 FPS higher than RefinedeLite, which was 7.633. The number of parameters is 7,060,349, which is about 2.4% smaller than YOLOv5.

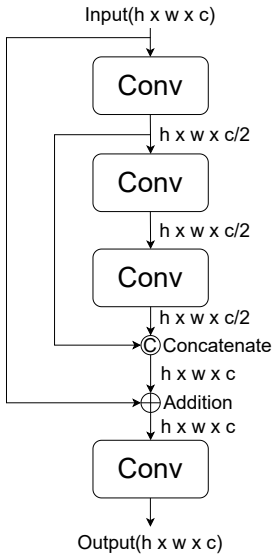

Fig. 2. Efficient Residual Bottleneck.

TABLE I
Detection result comparisons on MS COCO, where time@CPU1 means running time tested on Intel i7-6700@3.40GHz and time@CPU2 means running time tested on Intel i5 6600@3.30GHz.

Model	mAP 0.5:.95	Backbone	time@CPU1	time@CPU2
SSD [6]	0.193	MobileNet	128ms	-
SSDLite [7]	0.222	MobileNet	125ms	-
SSDLite [7]	0.221	MobileNetV2	120ms	-
Pelee [8]	0.224	PeleeNet	140ms	-
Tiny-DSOD [9]	0.232	DDB-Net+D-FPN	180ms	-
SSD [6]	0.251	VGG	1250ms	-
SSD [6]	0.28	ResNet101	1000ms	-
YOLOv3 [10]	0.282	DarkNet53	1300ms	-
RefineDetLite [11]	0.268	Res2NetLite72	130ms	-
RefineDetLite++ [11]	0.296	Res2NetLite72	131ms	-
YOLOv5s-ERB	**0.367**	**CSPDarknet53**	-	**43ms**
YOLOv5s-ERB_wosppf	**0.334**	**CSPDarknet53**	-	**36ms**
YOLOv5s-ERB_conv3	**0.366**	**CSPDarknet53**	-	**40ms**

Acknowledgment

This result was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-003)

References

[1] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "Cspnet: A new backbone that can enhance learning capability of cnn," 2019.

[2] "Yolov5," https://github.com/ultralytics/yolov5.

[3] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," *CoRR*, vol. abs/1803.01534, 2018. [Online]. Available: http://arxiv.org/abs/1803.01534

[4] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[5] M. D. Putro, D.-L. Nguyen, A. Priadana, and K.-H. Jo, "Fast person detector with efficient multi-level contextual block for supporting assistive robot," in *2022 IEEE 5th International Conference on Industrial Cyber-Physical Systems (ICPS)*, 2022, pp. 1–6.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015. [Online]. Available: http://arxiv.org/abs/1512.02325

[7] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation," *CoRR*, vol. abs/1801.04381, 2018. [Online]. Available: http://arxiv.org/abs/1801.04381

[8] R. J. Wang, X. Li, S. Ao, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," *CoRR*, vol. abs/1804.06882, 2018. [Online]. Available: http://arxiv.org/abs/1804.06882

[9] Y. Li, J. Li, W. Lin, and J. Li, "Tiny-dsod: Lightweight object detection for resource-restricted usages," *CoRR*, vol. abs/1807.11013, 2018. [Online]. Available: http://arxiv.org/abs/1807.11013

[10] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *CoRR*, vol. abs/1804.02767, 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[11] C. Chen, M. Liu, X. Meng, W. Xiao, and Q. Ju, "Refinedetlite: A lightweight one-stage object detection framework for cpu-only devices," *CoRR*, vol. abs/1911.08855, 2019. [Online]. Available: http://arxiv.org/abs/1911.08855