IEICE TRANS. INF. & SYST., VOL.E106–D, NO.5 MAY 2023

LETTER

Convolution Block Feature Addition Module (CBFAM) for Lightweight and Fast Object Detection on Non-GPU Devices

Min Ho KWAK[†], **Youngwoo KIM**[††], **Kangin LEE**[†††], ***and*** **Jae Young CHOI**[†a)], ***Nonmembers***

SUMMARY This letter proposes a novel lightweight deep learning object detector named LW-YOLOv4-tiny, which incorporates the convolution block feature addition module (CBFAM). The novelty of LW-YOLOv4-tiny is the use of channel-wise convolution and element-wise addition in the CBFAM instead of utilizing the concatenation of different feature maps. The model size and computation requirement are reduced by up to 16.9 Mbytes, 5.4 billion FLOPs (BFLOPS), and 11.3 FPS, which is 31.9%, 22.8%, and 30% smaller and faster than the most recent version of YOLOv4-tiny. From the MSCOCO2017 and PASCAL VOC2012 benchmarks, LW-YOLOv4-tiny achieved 40.2% and 69.3% mAP, respectively.

key words: *fast object detection, lightweight network, non-GPU devices*

1. Introduction

The existing deep network structures [1] present excellent object detection accuracy owing to their deeper and high-level feature extraction, which focuses on object location within an image. However, these methods involve complex network structures, and several network parameters must be learned. Consequently, they generally require a large amount of graphic processing unit (GPU) computation power. The application of these methods to non-GPU-based devices with limited computing power and memory size (e.g., augmented reality and mobile devices) remains challenging despite the extensive research conducted for the development of fast object detection methods [2]–[4]. Xu et al. [2] proposed an improved YOLOv3 model using DenseNet to improve the detection speed. They designed a lightweight deep network unit to replace the residual units in YOLOv3. Wong et al. [3] proposed the YOLO Nano model using a human-machine collaboration design approach. Zhao et al. [4] proposed a mixed YOLOv3-LITE model, which integrates various convolutional neural network (CNN) concepts, presenting a faster processing speed in an edge computing device.

This letter presents a novel LW(lightweight)-YOLOv4-tiny network based on the original YOLOv4-tiny model [5]. The novelty of the proposed LW-YOLOv4-tiny model lies in the development of the convolution block feature addition module (CBFAM), which performs channel-wise convolution and element-wise addition instead of the conventional concatenation of feature maps. CBFAM reduces the computational complexity and the parameters to increase the speed of the network. Additionally, the proposed LW-YOLOv4-tiny model achieves a balance between detection speed and accuracy by adjusting the design parameters, as shown in Eq. (1). The LW-YOLOv4-tiny model presents a much faster object detection speed than the original YOLOv4-tiny model, based on results of the experiments conducted on the MSCOCO2017 [6] and PASCAL VOC 2007 [7] benchmarks [5]. Additionally, it achieves better detection accuracy than the recently developed lightweight object detector models.

2. Overview

Figure 1 depicts the overall structure of the proposed LW-YOLOv4-tiny network for fast and accurate object detection. "Conv" represents a convolutional layer followed by a leaky ReLU activation function. Figure 2 presents the detailed structure of the CBFAM. In both Figs. 1 and 2, D[n, m] stands for removing the channels of our feature map of the final CBFAM indexed from n to m, aiming to match different channel size when performing element-wise addition. Note that in Fig. 1, there are two outputs of the final CBFAM – one (indicated by dot line at the right side) results from the Leaky activation output, while the other (indicated

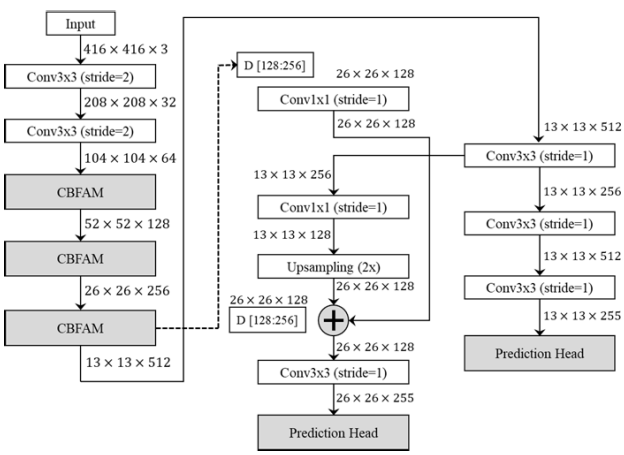
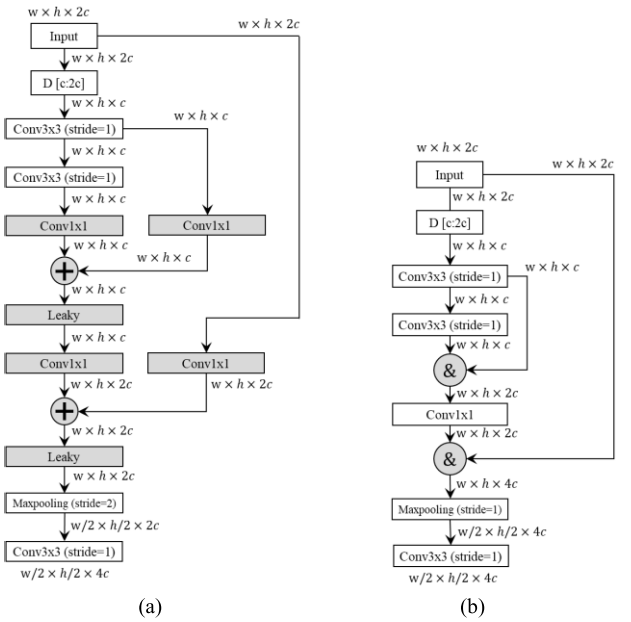


Fig. 1 Proposed LW-YOLOv4-tiny network structure; 'x × y × z' indicates the dimensions of the width, height, and channel, respectively, and ⊕ represents the element-wise addition.

Manuscript received December 21, 2022.
Manuscript publicized January 24, 2023.

[†]The authors are with Division of Computer Engineering, Hankuk University of Foreign Studies, Republic of Korea.
[††]The author is with Sustainable Technology and Wellness R&D Group, Korea Institute of Industrial Technology, Republic of Korea.
[†††]The author is with Institute of Advanced Technology Development, Hyundai Motor Company, Republic of Korea.
a) E-mail: jychoi@hufs.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2022EDL8104



(a) (b)

Fig. 2 Comparison between (a) the proposed CBFAM structure and (b) convolutional block of original YOLOv4-tiny model; ⊕ and & represent the element-wise addition and the concatenation, respectively. Note that the final "Conv3x3" layer should be included within CBFAM and CSP blocks and also the number of input channels becomes double in the final "Conv3x3" layer of each CBFAM.

by solid line at the bottom side) is formed with the inclusion of the last two components of the final CBFAM (i.e., "Maxpooling" and "Conv3x3" layers) (See Fig. 2 (a) for details).

The CBFAM comprises a series of convolution layers that integrate the operations of channel-wise convolutions, an activation function, and element-wise addition, as shown in Fig. 2. The concatenating operations are replaced with simple element-wise addition to increase the speed of the detection process. However, element-wise additions can reduce the detection accuracy because they do not account for the correlations between the channels of two different feature maps which are considered as the input. Therefore, channel-wise convolution layers are applied before performing element-wise addition. Furthermore, the detection performance deteriorates when the element-wise addition of two inputs is performed without using the activation function because the distribution of element-wise addition results may differ from the concatenation operations. A leaky rectified linear unit (Leaky ReLU) activation function is added at the end of the element-wise addition to solve this problem, to ensure that the result of the element-wise addition imitates the distribution of the concatenation operations, as shown in Fig. 2.

Note that the proposed CBFAM structure could effectively reduce the number of computations for object detection. The underlying reason is that the element-wise addition in the CBFAM does not increase the dimensionality of feature maps, which is different from the case of using concatenation. In addition, our combined use of 3x3 convolutional and channel-wise layers with reduced input channel size from the CBFAM has fewer weights than a single convolutional layer with a whole input channel from the original YOLOv4-tiny.

3. Convolution Block Feature Addition Module

3.1 Motivation

The original YOLOv4-tiny [5] model implemented a large number of concatenation operations, where the outputs from two different input feature maps are first concatenated and the resultant concatenated information is transmitted to the next convolution layer. The gradient information of both the current convolution layer and that of the previous layer must be used to update their weights. The concatenation operations of the original YOLOv4-tiny model improve the accuracy but increase the complexity of the network. Furthermore, the large size of the input channel adds several parameters to the convolution layer, which considerably increases the size of the model and requires more computational power. These disadvantages can be overcome by using systems which can perform parallel processing, such as a GPU. However, the systems which are not equipped with the GPU hardware can be substantially slowed down owing to a series of concatenation operations. This work employs channel-wise convolution layers and element-wise addition with an activation function instead of concatenating operations to overcome this issue.

3.2 Proposed CBFAM

Each convolutional block of the original YOLOv4-tiny model has two concatenation operations. In the proposed method, the current concatenation operations are replaced by the proposed CBFAM, which performs channel-wise convolution and addition (see Fig. 2). The proposed CBFAM is a modified version of residual connection [5] because the main branch of CBFAM utilizes only half of the input channels. This main branch utilizes a part of information when passing through a deeper level of the network for better generalization performance; this mechanism allows the main branch, which generates a high-level feature map, to utilize less information, and the residual connection branch, which generates relatively low-level feature map, to handle the whole information.

The CBFAM is developed to ensure that the existing concatenation operation adds several parameters to the convolution layers owing to its large number of concatenated channels, which significantly increases the number of I/O operations. Non-GPU (CPU) systems are not suitable for parallel processing and can be significantly slowed down by a series of concatenation operations because of the overwhelming volume of the I/O process. However, the proposed CBFAM block can significantly reduce the number of parameters of all the available convolution layers. Specifically, considering the kernel size, input and output channel sizes, and bias for each convolution layer, the num-

ber of parameters are calculated as 119 KB and 188 KB for the CBFAM and original CSP blocks [5], respectively, which reduces the number of parameters by up to 36.8% smaller than original CSP block. Such parameter reduction is the virtue of adopting the channel-wise convolution and element-wise addition in the proposed CBFAM blocks. In our CBFAM, concatenation operations were replaced with simple element-wise addition. Note that the element-wise addition does not increase the dimensionality of feature maps, which is contrast to the case of using concatenation operations, leading to a reduced number of output channels of CBFAM compared to the original YOLOv4-tiny model. Furthermore, the channel-wise convolutional layer allows model to understand the correlations between the channels before the inputs are added. The activation function is also applied to exploit the correlation between the channels of the two inputs.

The input of each CBFAM is defined as follows:

$$F_{cw}^{(n)}[0,\alpha] = Conv_{1x1}\left(F_{map}^{(n)}\right) \tag{1}$$

where $F_{map}^{(n)}$ represents the n-th convolution layer output in the CBFAM, and $Conv_{1x1}$ performs channel-wise convolution with a 1x1 size kernel and $F_{cw}^{(n)}[0,\alpha]$ is obtained as a result by applying $F_{map}^{(n)}$ to the channel-wise convolution computation. Here, α represents a parameter that determines the number of channels used to form $F_{cw}^{(n)}[0,\alpha]$ by selecting a part of the entire channel of $F_{map}^{(n)}$. Essentially, $F_{cw}^{(n)}[0,\alpha]$ consists of the selected channels of $F_{map}^{(n)}$ ranging from the 1st channel to the $\left(|F_{map}^{(n)}| * \alpha\right)^{th}$ channel, where $|\ |$ denotes the cardinality of the set. For instance, if α and the total number of channels of $F_{map}^{(n)}$ are 0.5 and 256, respectively, only the channels from the 1st to 128^{th} order can be selected to construct $F_{cw}^{(n)}[0,\alpha]$. An optimal balance can be achieved between the model size, detection speed, and detection accuracy by adjusting α (see Fig. 3).

To produce the CBFAM output, the intermediate output

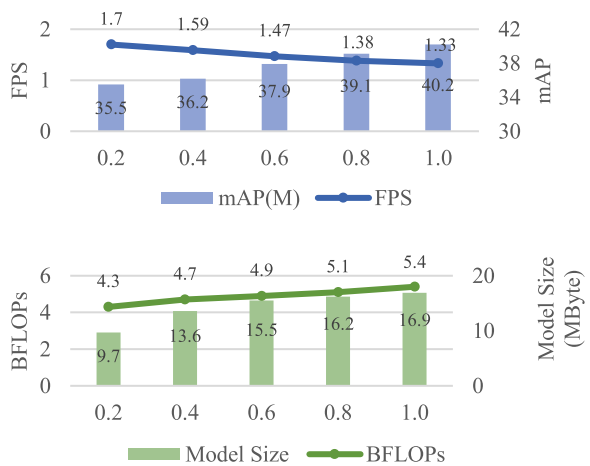


Fig. 3 Design flexibility of the LW-YOLOv4-tiny model in terms of detection accuracy and speed as well as model size by varying α in Eq. (1); MSCOCO2017 test set and ARM Cortex A57 CPU were used.

can be written as follows:

$$O_{im} = Leaky\left(F_{cw}^{(n-1)}[0,\alpha] \oplus F_{cw}^{(n-2)}[0,\alpha]\right) \tag{2}$$

where $F_{cw}^{(n-1)}[0,\alpha]$ and $F_{cw}^{(n-2)}[0,\alpha]$ represent the $(n-1)^{th}$ and $(n-2)^{th}$ convolution layer outputs of the CBFAM, respectively, which are obtained as described in Eq. (1), "$\oplus$" denotes element-wise addition, and *Leaky* is the Leaky ReLU activation function [5]. This structure is a type of skip connection [5] which learns residual functions corresponding to other layer inputs instead of learning unreferenced functions. A deep network based on this structure is easier to optimize and can gain accuracy with a considerably increased depth. The final output of each CBFAM can be expressed as follows:

$$\tilde{O}_{im} = Leaky(Conv_{1x1}(O_{im}) \oplus F_{cw}^{(n)}[0,\alpha]) \tag{3}$$

and

$$Output = Conv3x3(Maxpooling(\tilde{O}_{im}). \tag{4}$$

The *Output* in Eq. (4) represents the final output of the CBFAM, computed via the element-wise addition of O_{im} and $F_{cw}^{(n)}[0,\alpha]$ with a leaky activation.

4. Experiments and Results

The proposed LW-YOLOv4-tiny model is implemented based on the CSPDarknet53-tiny backbone network [5] available in the Darknet framework [9]. The cross-stage residual connection [4] of the original CSPDarknet53-tiny model is replaced by the proposed CBFAM. Comparative experiments were performed using MSCOCO2017 [6] and PASCALVOC2012 [7]. MSCOCO2017 has 123,287 RGB images with 886,284 instances, and PASCAL VOC 2012 has 21493 images. The same evaluation setup [7] is employed for PASCALVOC2012, using the training set (21493 images) and the test set (4952 images). The detection accuracy, i.e., mean average precision (mAP), is measured by using MSCOCO2017 testdev [6] on the detection challenge server CodaLab to obtain a fair comparison [8]. The ADAM [11] is used for training, with the batch size of 64. The learning rate starts with 1e-4 and is multiplied by 0.1 after the iteration reaches at 40000, 60000, and 80000. The complexity of the model is evaluated by using billion FLOPs (BFLOPs), model size, and number of parameters. All models used 416×416 input size to get consistent evaluations. Also note that α shown in Eq. (1), (2), and (3) was set to '1' in all of the following experiments, if not stated otherwise.

Table 1 presents the results of the evaluation with 3.8 GHz Intel Core i7 CPU and a single NVIDIA RTX 3090 GPU. During the test, only the CPU is used for the inference. According to the test results, the mAP of the LW-YOLOv4-tiny model is observed to be clearly higher than those of the YOLOv4-tiny and YOLO Nano series networks, and it presents better performance in terms of the amount of computation and model size (the number of parameters), as shown in Table 1. Particularly, the model size and BFLOPs

Table 1 Evaluation results for LW-YOLOv4-tiny and other state-of-the-art lightweight models. The $mAP^{(M)}$ and $mAP^{(P)}$ denote mAPs on MSCOCO2017 and PASCALVOC. The α in Eq. (1) was set to 1.

Model	$mAP^{(M)}$	$mAP^{(P)}$	Size (MB)	BFLOPs
YOLOv3-tiny [10]	33.8	60.6	33.8	5.6
YOLOv4-tiny [5]	40.2	68.8	24.3	6.9
Tiny YOLOv3 LITE [4]	37.4	72.9	20.1	5.0
YOLO Nano [3]	36.9	69.1	4.0	4.6
LW-YOLOv4-tiny	40.2	69.3	16.9	5.4

Table 2 Comparison of detection speed (FPS) for different CPU-based embedded devices with Ubuntu 20.0.4 operating system and 32-bit Floating Point (FP32) operations. MSCOCO2017 testdev was used. The α in Eq. (1) was set to 1.

CPU Type / Model	Intel i5-8265U	Intel Cortex A57	vCPU x2 (Google Cloud)	vCPU x4 (Google Cloud)
Tiny YOLOv3 LITE [4]	9.3	1.28	2.83	5.61
YOLO Nano[3]	9.3	0.94	2.16	5.96
YOLOv4- tiny [5]	8.42	0.98	2.52	5.07
LW YOLOv4-tiny	11.3	1.33	3.05	6.10

Table 3 Ablation studies of essential components in CBFAMs; MSCOCO2017 testdev was used.

LW-YOLOv4-tiny		mAP	BFLOPs	Size (MB)
Element-wise addition	Channel-wise convolution			
X	X	40.2	6.9	24.3
O	X	36.8	4.9	14.9
X	O	40.6	8.5	28.4
O	O	40.2	5.4	16.9

*For channel selection, the α in Eq. (1) was set to 1.

are reduced by approximately 30% and 22%, respectively, while the mAP of LW-YOLOv4 is equivalent or even better when compared to the original YOLOv4-tiny model.

The proposed LW-YOLOv4-tiny model is tested under different CPU (non-GPU)-based devices, which employ a low-performance computing system with low power consumption. Table 2 presents the results. The proposed LW-YOLOv4-tiny model had a higher FPS than the other models. Remarkably, the proposed LW-YOLOv4-tiny model presents a better FPS when compared to the original YOLOv4-tiny model, by up to 11.3 and 1.33 on Intel i5-8265U and ARM Cortex A57, respectively.

The LW-YOLOv4-tiny model achieves a much better processing speed for the Google Cloud environment with the N2 series platform, by up to 20% and 21% for vCPU x2 and x4, respectively. The primary reason for faster detection of the proposed CBFAM is as follows: (a) the number of parameters used during the convolution process can be considerably reduced by using channel-wise convolution and (b) the number of input channels for all convolution layers should be decreased by adopting element-wise addition instead of using concatenation operations.

Table 3 shows the results of ablation studies to validate the effectiveness of essential components such as channel-wise convolution and element-wise addition in our CBFAMs. We can see that by combining both channel-wise convolution and element-wise addition in CBFAMs, our proposed LW-YOLOv4-tiny attains the best performance compared to other variants considering the balance between detection accuracy, lower computational complexity, and memory size. Figure 3 depicts the design flexibility of the LW-YOLOv4-tiny model in terms of controlling the trade-off between the model size, detection accuracy, and speed by adjusting the parameter, α, in Eq. (1). A good compromise was found by setting α in the range of 0.8 and 1, considering both the computational complexity and detection performance, based on the results shown in Fig. 3.

5. Conclusion

The proposed LW-YOLOv4-tiny presents a faster detection speed and almost identical detection accuracy when compared to other state-of-the-art lightweight networks, particularly when non-GPU-based embedded devices are implemented.

Acknowledgements

This work was supported by the Hankuk University Foreign Studies Research Fund. This work was also supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2021R1A2C1092322). This work was partially supported by the "Development of Wave Overtopping quantitative observation technology" funded by the Korea Institute of Marine Science & Technology Promotion (KIMST) (No. 20220180).

References

[1] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," arXiv 2018, arXiv preprint arXiv:1809.02165, 2019.

[2] Z.-F. Xu, R.-S. Jia, H.-M. Sun, Q.-M. Liu, and Z. Cui, "Light-YOLOv3: Fast method for detecting green mangoes in complex scenes using picking robots," Applied Intell., vol.50, no.12, pp.4670–4687, 2020.

[3] A. Wong, M. Famuori, M.J. Shafiee, F. Li, B. Chwyl, and J. Chung, "Yolo nano: A highly compact you only look once convolutional neural network for object detection," arXiv preprint arXiv:1910.01271, 2019.

[4] H. Zhao, Y. Zhou, L. Zhang, Y. Peng, X. Hu, H. Peng, and X. Cai, "Mixed YOLOv3-LITE: A lightweight real-time object detection method," Sensors, vol.20, no.7, 1861, 2020.

[5] A. Bochkovskiy, C.Y. Wang, and H.-Y.M. Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

[6] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C.L. Zitnick, "Microsoft COCO: Common objects in context," European Conference on Computer Vision, Lecture Notes in Computer Science, vol.8693, pp.740–755, Springer, Cham., Sept. 2014.

[7] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, and A.

Zisserman, "The PASCAL visual object classes (VOC) challenge," International Journal of Computer Vision, vol.88, no.2, pp.303–338, 2010.

[8] COCO Detection Challenge (Bounding Box), https://competitions.codalab.org/competitions/20794, accessed June 2021.

[9] A. Bochkovskiy, "Yolo v4, v3 and v2 for Windows and Linux," https://github.com/AlexeyAB/darknet, accessed Dec. 2020.

[10] A. Farhadi and J. Redmon, "yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.

[11] D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.