Efficient Detection Model using Feature Maximizer Convolution for Edge Computing*

Jehwan Choi[1][0009−0005−8494−2170], Youlkyeong Lee[2][0000−0002−5806−7502], and Kanghyun Jo[3][0000−0001−8317−6092]

Department of Electrical, Electronic and Computer Engineering,
University of Ulsan, Ulsan, South Korea
http://islab.ulsan.ac.kr
{cjh1897[1], yklee[2], acejo[3]}@ulsan.ac.kr

Abstract. Deep learning is heavily influenced by the quantity and quality of data. Moreover, most deep learning models are developed and experimented on servers equipped with high-performance GPUs and large memory capacities. However, for practical application in industrial fields or real-world scenarios, optimal performance must be achieved with limited resources and equipment. Therefore, this paper proposes the Feature Maximizer Convolution (FMC), which aims to enhance performance with limited data by extracting as diverse features as possible and assigning more weight to feature maps containing crucial information, which are then passed on to the next layer. Additionally, to ensure real-time performance on limited hardware, Depthwise Separable Convolution is employed instead of standard convolution to reduce computational load. This approach is applied to deep learning models on datasets such as COCO, VisDrone, VOC, and xView, comparing performance with existing networks, and inference experiments are also conducted on the edge device Odroid H3+. The proposed network shows a 30% average reduction in the number of parameters compared to existing networks, and a 5% decrease in inference speed. On the Odroid H3+, the inference speed improved by an average of 2.5ms, increasing from 19FPS to 20FPS.

Keywords: Computer vision · deep learning · edge computing · lightweight.

1 Introduction

As deep learning rapidly evolves, it has begun to be applied across various industries. A typical deep learning process involves extracting features from data through Convolution operations, computing these features to produce probabilistic outcomes, and then determining the most probable formula by comparing these outcomes with the Ground Truth. This process signifies that the deep learning process is influenced by the accuracy and characteristics of the data.

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

Consequently, in recent years, many countries have been focusing on acquiring and constructing data, and research into learning methods such as Data Augmentation [1, 2], semi-supervised learning [3], and Few-shot learning [4] that yield good results with limited data has been extensive. Moreover, from the perspective of computer vision, object detection, which deals with various datasets (COCO [5], VOC [6], xView [7], VisDrone [8], etc.), is a significant challenge. These datasets include objects of various appearances and sizes, from those that are clearly visible to those that are hardly discernible due to high altitude or distance. Therefore, it is common to use pre-trained models on various datasets and apply transfer learning according to specific conditions and tasks. However, to enhance the performance of transfer learning, the pre-trained model itself must also be high-performing. To ensure good performance in any task, a network capable of effectively extracting features of various sizes and shapes is necessary. Faster R-CNN improved accuracy by quickly generating candidate regions for objects using the Region Proposal Network (RPN) [9], while the SSD (Single Shot MultiBox Detector) [10] and YOLO (You Only Look Once) [11–13] series demonstrated good performance on objects of various sizes using feature maps of different scales. In this paper, we propose the FMC (Feature Maximizer Convolution) module, which maximizes the diversity of limited features and effectively extracts features through computations among feature maps, utilizing superior feature maps.

As mentioned earlier, extracting features effectively from data and the quantity and quality of data are crucial for enhancing the performance of deep learning. However, the amount and quality of data are expected to increase over time, leading to improved learning outcomes. Yet, deploying these data-driven models in real-world applications presents another challenge. This is because inference using servers with high-performance GPUs and large memory capacities is difficult to apply in actual industries. In reality, devices such as smartphones, autonomous vehicles, and CCTV cameras do not have servers equipped with high-performance GPUs or large memory capacities. Therefore, to use deep learning efficiently in real-world scenarios, it is necessary to achieve good performance with limited power and hardware specifications. This task is referred to as Edge Computing or Edge AI. With the growing trend of edge computing, there is an increasing need for lightweight yet powerful models that can operate efficiently on edge devices. Recent research focuses on enhancing the practicality and efficiency of deep learning technologies in Edge Computing and Edge AI. For example, Lian et al. (2023) [14] proposed a new method for testing and strengthening the vulnerability of deep learning models in Edge Computing environments. This suggests that Edge AI can play a significant role in enhancing reliability and efficiency in real-world settings. Additionally, Zeng et al. (2023) [15] developed a new task scheduling algorithm for efficiently allocating large and dynamic workloads on Edge Computing nodes. These studies contribute significantly to improving the efficiency and performance of Edge AI. In this paper, we propose applying DSC (Depth-wise Separable Convolution) instead of standard convolution operations to the main modules of the Object

Detection model YOLOv5, reducing computational load for efficient operation on Edge Devices. We also introduce experimental results showing the improved performance of the proposed model implemented on the Edge Device Odroid H3+ compared to existing models. The main contributions of this article are summarized as follows:

- We propose the FMC (Feature Maximizer Convolution) module to extract as diverse features as possible from limited feature maps and to use only the most effective features.
- We implement model lightening by applying DSC (Depth-wise Separable Convolution) instead of standard convolution operations in the main modules of YOLOv5.
- We conduct experiments and analyze the performance changes compared to existing models using the Edge device Odroid H3+.

2 Related Work

2.1 Edge Computing

Edge Computing (EC) is a technology that can overcome the limitations of Cloud Computing (CC). The typical limitations of cloud computing include: a. the latency involved in sending data to a server capable of computation, processing it, and then receiving the results, b. the increased costs and network congestion due to the high bandwidth required to transfer large volumes of data to the cloud, and c. cyber security issues that can arise during data transmission. The core technologies of EC to solve these problems include: a. real-time response by processing data immediately at the site of data generation, b. distributed processing that reduces system load and increases efficiency by distributing data across multiple edge devices, and c. enhanced security by processing data on-site, thereby protecting against data breaches and hacking. In terms of recent research in computer vision, Guanchu Wang et al. (2022) [16] demonstrated a system called BED (oBject detection system for Edge Devices), which creates a small DNN (Deep Neural Network) model of 300kb through an end-to-end pipeline of model training, quantization, synthesis, and deployment. Shihan Liu et al. (2023) [17] achieved results of over 30FPS on Nvidia jetson AGX Xavier using Data Augmentation with Mosaic and an Efficient Decoupled Head. This paper also improves results by efficiently adjusting the feature map.

2.2 Efficient Feature Extraction

Efficient feature extraction is a crucial research topic in the fields of deep learning and computer vision, particularly in real-time image processing and object detection on edge devices with limited computing resources. Recent studies focus on maintaining high performance while reducing model complexity. Notably, in the MobileNet series by Andrew G. Howard et al. (2017) [18], techniques like DSC (Depth-wise Separable Convolution) were used to reduce the number

of parameters and increase computational efficiency. Tsung-Yi Lin et al. (2017) proposed the FPN (Feature Pyramid Network) [19], extracting feature maps of various scales to achieve significant results in detecting small objects. The reason for applying these methods is to achieve good performance and fast inference speed in limited environments and data. Reducing computational load while maintaining object detection performance can speed up inference, ensuring real-time processing. Conversely, improving object detection performance while maintaining computational load allows for more effective application of models in real life and industry. The DSC used in this paper can be seen in Figure 1.

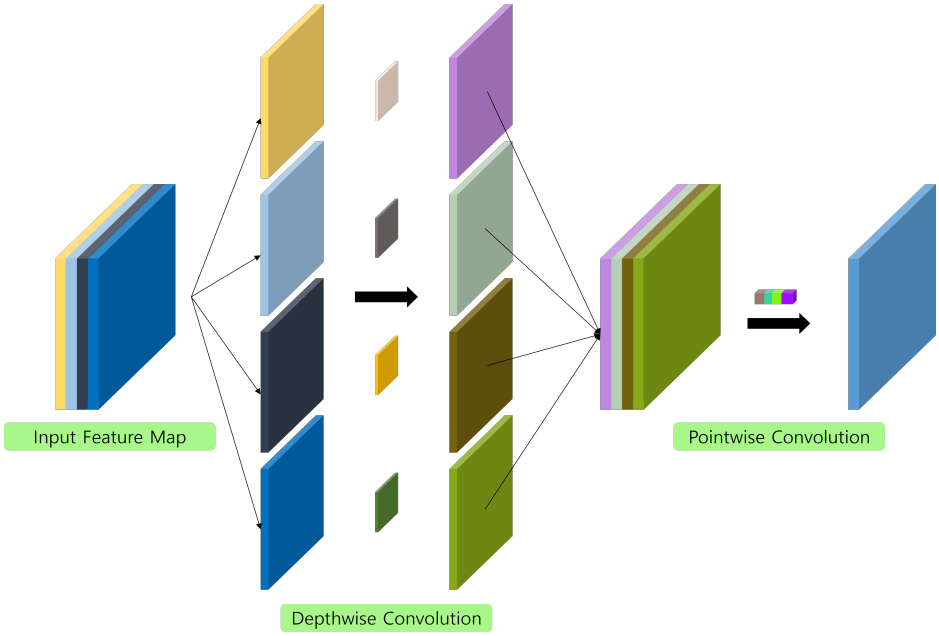


Fig. 1: The Computational Process of Depthwise Separable Convolution (DSC)

3 Proposed Method

In this paper, YOLOv5 is used as the base model. Instead of the core C3 module of YOLOv5, the FMC (Feature Maximizer Convolution) module is used to extract more diverse features from the feature map. To reduce computational load, Depthwise Separable Convolution (DSC) is employed instead of standard convolution operations.

3.1 Feature Maximizer Convolution

In this paper, the C3 module used in YOLOv5, which consists of multiple layers of convolution operations, is discussed. The C3 module is based on the CSP (Cross Stage Partial) structure. This structure divides the input feature map into two parts, applies convolution operations intensively to one part, and then merges them back together. However, most of these operations are carried out using 1x1 convolutions. While 1x1 convolutions have the advantage of combining features across channels and reducing dimensions with less computational load, they are limited in capturing spatial information. To address this, the FMC (Feature Maximizer Convolution) is proposed in this paper. Like the C3 module, the FMC also consists of two branches. The first branch applies a 3x3 convolution to the input feature map to compensate for the shortcomings of 1x1 convolutions, but uses DSC (Depthwise Separable Convolution) to reduce computational load. It does not reduce the channel dimension to retain as diverse features as possible. Then, the feature map processed by 3x3 DSC and the input feature map are concatenated, followed by a 1x1 convolution to adjust the weights for extracting important features. The second branch applies another 3x3 DSC to the feature map processed by the first branch's 3x3 DSC. Finally, the first and second branches are concatenated to output the final feature map. This configuration is designed to generate as diverse feature maps as possible and extract features that effectively enhance performance with limited data. The C3 and FMC modules are illustrated in Figure 2.

3.2 Lightweight Strategy

The existing YOLOv5 pipeline consists of a sequence of C3 and Conv modules. As seen in Figure 2, the C3 module predominantly applies 1x1 convolutions. As shown in Equation (1), 1x1 convolution is the operation with the least number of parameters among standard convolution operations. Therefore, as indicated in Equation (2), applying DSC (Depthwise Separable Convolution) with the same kernel size increases the number of parameters by the number of input channels.

$$k * k * c_{\text{in}} * c_{\text{out}} = 1 * 1 * c_{\text{in}} * c_{\text{out}} = c_{\text{in}} * c_{\text{out}} \tag{1}$$

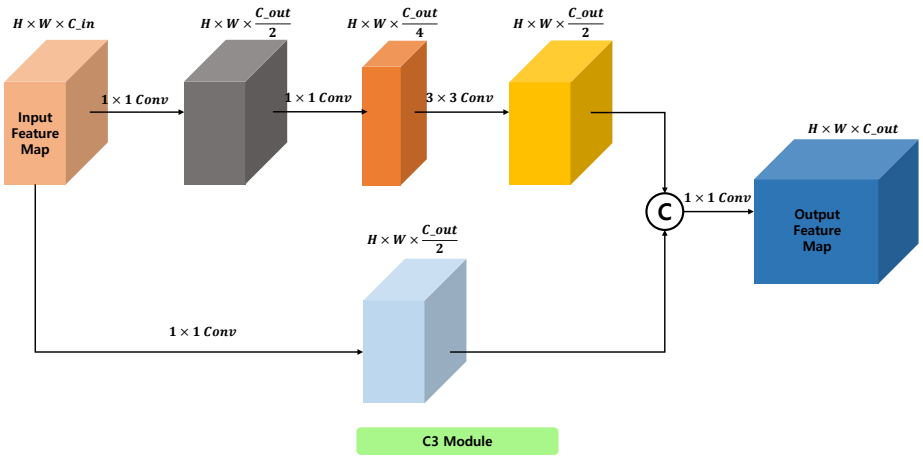
$$k * k * c_{\text{in}} + 1 * 1 * c_{\text{in}} * c_{\text{out}} = 1 * 1 * c_{\text{in}} + 1 * 1 * c_{\text{in}} * c_{\text{out}} = c_{\text{in}}(1 + c_{\text{out}}) \tag{2}$$

k represents the kernel size, while c_{in} and c_{out} denote the number of input and output channels, respectively. As per the above formula, the number of parameters for the C3 and FMC modules can be calculated as shown in Equations 3 and 4, respectively.

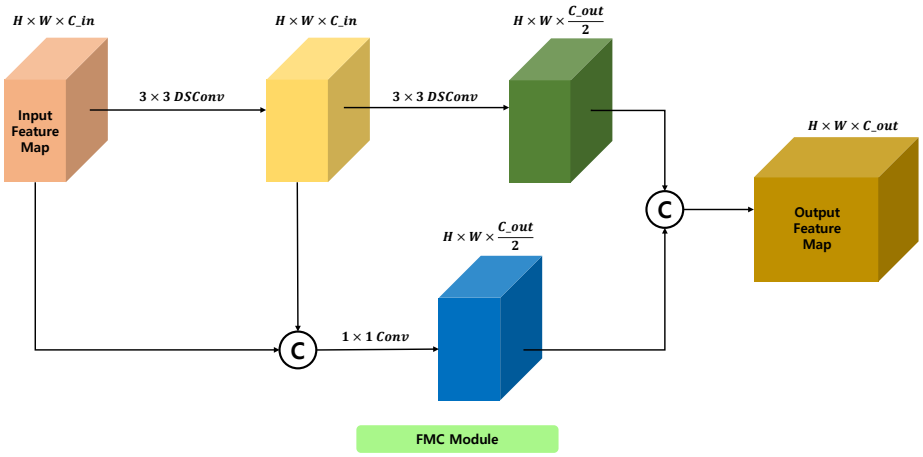
$$c_{\text{out}}(c_{\text{in}} + 2.25 * c_{\text{out}}) \tag{3}$$

$$c_{\text{in}}(18 + c_{\text{in}} + 1.5 * c_{\text{out}}) \tag{4}$$

To compare Equations (3) and (4), let's assume that the number of input and output channels are the same ($c_{\text{in}} = c_{\text{out}} = C$). As a result, the number of parameters for the C3 module is $3.25C^2$, and for the FMC module, it is $2.5C^2 +$



(a) C3 module used in YOLOv5



(b) Proposed Feature Maximizer Convolution (FMC) module

Fig. 2: Images of the computational processes of the C3 and FMC modules

$18C$. Upon comparison, it is evident that the C3 module has more parameters when $C > 24$, and the FMC module has more when $C < 24$. Since most deep learning networks typically have more than 32 channels, it can be inferred that the FMC module has fewer parameters than the C3 module.

The Conv module involves standard convolution operations followed by batch normalization and an activation function. However, in YOLOv5, most Conv operations have a kernel size of 3. If k is applied as 3 in Equations (1) and (2), standard convolution is calculated as shown in Equation (5), and DSC is calculated as shown in Equation (6), indicating that DSC has fewer parameters. Therefore, in this paper, all Conv modules in the YOLOv5 Backbone have been modified to DSC modules for model lightening.

$$k * k * c_{\text{in}} * c_{\text{out}} = 3 * 3 * c_{\text{in}} * c_{\text{out}} = 9 * c_{\text{in}} * c_{\text{out}} \tag{5}$$

$$k * k * c_{\text{in}} + 1 * 1 * c_{\text{in}} * c_{\text{out}} = 3 * 3 * c_{\text{in}} + 1 * 1 * c_{\text{in}} * c_{\text{out}} = c_{\text{in}}(9 + c_{\text{out}}) \tag{6}$$

Fig. 3: The result image of xView dataset

4 Experiment

4.1 Dataset

COCO: The COCO dataset, short for 'Common Objects in Context', contains a variety of objects found in everyday life. It consists of approximately 200,000 images and includes 80 different classes. COCO is widely used for various computer vision tasks such as object detection, segmentation, and keypoint detection, and is characterized by its complex backgrounds and a range of object sizes.
VOC: The PASCAL VOC (Visual Object Classes) dataset is a benchmark

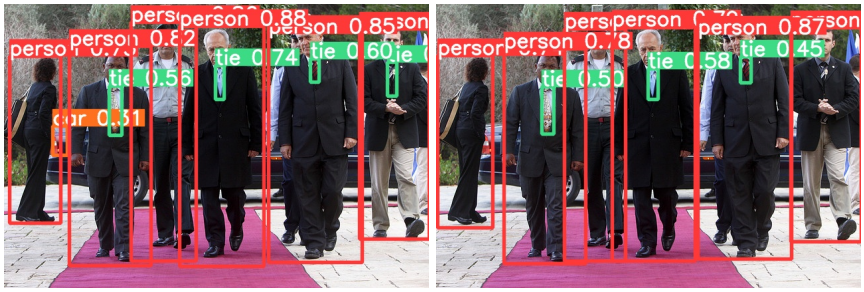


Fig. 4: The result of big size human, (Left) YOLOv5n (Right) Proposed method

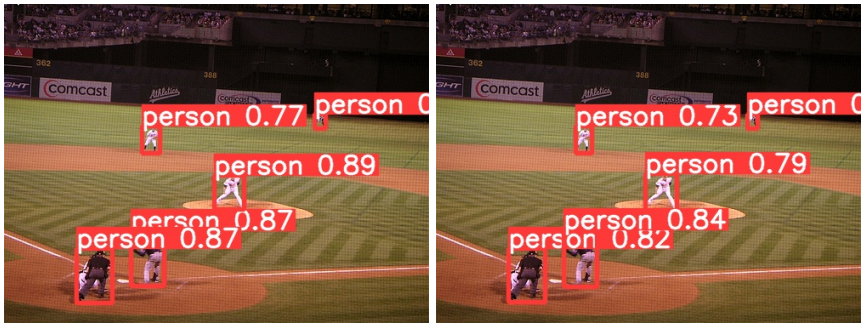


Fig. 5: The result of small size human, (Left) YOLOv5n (Right) Proposed method

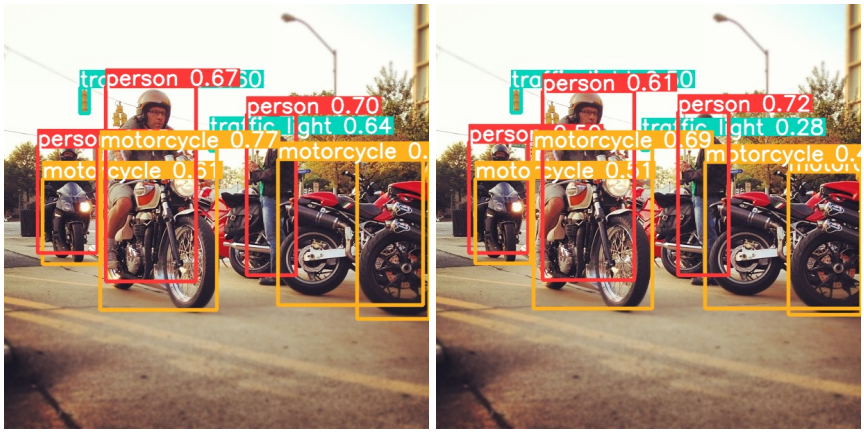


Fig. 6: The result of various object, (Left) YOLOv5n (Right) Proposed method

Fig. 7: The result of many objects, (Left) YOLOv5n (Right) Proposed method

dataset for object detection, image classification, and object segmentation. It includes 20 object categories and has evolved through the VOC challenge. Its main features and challenges are the variety of object sizes, overlapping objects, and diverse poses and backgrounds.

VisDrone: The VisDrone dataset is based on images and videos captured by drones for object detection and tracking. It includes high-resolution images and videos shot in various urban, rural, and coastal areas. The main challenges of this dataset are small objects captured from high altitudes, varying lighting conditions, and blur caused by camera movement.

xView: The xView dataset is for object detection based on satellite imagery. It contains over one million object instances and more than 60 object categories. Providing data captured in various geographical locations through high-resolution satellite imagery, its main challenges are detecting small objects and handling objects of various scales.

4.2 Evaluation Metric

For the evaluation metric, we measure the number of parameters, which affects the computational speed, and FLOPs (Floating Point Operations Per Second). Additionally, to assess the object detection accuracy of the generated model, we set mAP50 and mAP50-95 as indicators. Finally, to determine if the conditions are suitable for operation on Edge Devices, we compare the size of the generated model (in Mb) and the inference speed.

Table 1: Comparison of YOLOv5 and Proposed Network Performance Results on Four Datasets

Dataset	Model / Result	YOLOv5 nano	Proposed nano	Change
COCO	Parameters	1,867,405	1,309,673	-29.87%
	GFLOPs	4.5	3.2	-28.89%
	mAP50	40.9	35.3	-13.69%
	mAP50-95	24	20.2	-15.83%
VOC	Parameters	1,786,225	1,228,493	-31.22%
	GFLOPs	4.2	2.9	-30.95%
	mAP50	64.2	62.5	-2.65%
	mAP50-95	37.5	35.7%	-4.8%
VisDrone	Parameters	1,772,695	1,214,963	-31.46%
	GFLOPs	4.2	2.9	-30.95%
	mAP50	18.2	16.1	-11.54%
	mAP50-95	8.77	7.11	-18.93%
xView	Parameters	1,840,345	1,282,613	-30.31%
	GFLOPs	4.4	3.1	-29.55%
	mAP50	0.141	0.143	+1.42%
	mAP50-95	0.0465	0.0474	+1.94%

4.3 Experimental Setting

All training was conducted under the same environment and conditions, and the model training was carried out on a server equipped with an Intel Core i9-9960X, Nvidia RTX 2080 Ti x 4EA, and 125.5 GB of memory. The best-performing results out of 100 epochs were used for all models. Model performance evaluation was conducted on both the training server and the edge device, Odroid H3+.

Table 2: Comparison of YOLOv5 and Proposed Network Performance Results on Four Datasets

Dataset	Model / Result	YOLOv5 nano	Proposed nano	Change
COCO	Model size(Mb)	3.82	2.74	-28.27%
	Inference time(ms/image)	55.21	53.12	-3.79%
VOC	Model size(Mb)	3.68	2.61	-29.08%
	Inference time(ms/image)	50.94	47.67	-6.42%
VisDrone	Model size(Mb)	3.65	2.57	-29.59%
	Inference time(ms/image)	42.38	40.42	-4.62%
xView	Model size(Mb)	3.87	2.79	-27.91%
	Inference time(ms/image)	62.28	59.62	-4.27%

4.4 Result

The experiment was conducted using four datasets on two networks: the original YOLOv5 nano model and a nano model with the modules proposed in the paper. The results for Parameters, GFLOPs, mAP50, and mAP50-95 tested on the training server can be found in Table 1. It is observed that the number of parameters and computational load of the proposed model is reduced by about 30% compared to the original model. However, the reduction in computational load leads to a decrease in object detection accuracy, ranging from about 3% to as much as 19%. In the case of the xView dataset, the accuracy actually increased. As shown in Figure 3, considering that the images are satellite photos and most objects in the dataset's 60+ classes are concentrated in buildings, this appears to be an exceptional result with significantly reduced accuracy.

The results of inference on the Odroid H3+ using the same models and datasets as in Table 1 can be found in Table 2. Parameters, GFLOPs, and accuracy are omitted as they showed no significant deviation from Table 1, and the comparison focuses on model size and average inference speed per image. With the reduction in the number of parameters and GFLOPs, the model size also decreased by an average of about 29%. The inference time for the original model was already fast, showing an average performance of about 19FPS with 50-60ms.

Although the proposed model showed a relatively small reduction in inference time compared to the decrease in parameters and model size, it demonstrated a performance improvement of about 1FPS, averaging 20FPS with 45-55ms.

5 Conclusion

"This paper investigates methods to achieve good performance with limited environments and data, and studies the impact of applying a deep learning model with the proposed methods on edge devices in real-world settings. We propose the Feature Maximizer Convolution (FMC) to extract as diverse features as possible from the given data, ensuring that features beneficial for performance enhancement are passed on to the next layer. FMC is applied in place of the C3 module in YOLOv5. Additionally, to maintain performance while reducing computational load, Depthwise Separable Convolution (DSC) is applied. DSC replaces the Conv module in YOLOv5, significantly reducing computational load while maintaining similar performance to 3x3 convolutions. The original YOLOv5-trained model's nano version averages 3.755 Mb, but the model trained with the proposed method averages 2.7Mb, a reduction of about 28.1%. On the edge device Odroid H3+, the inference time also improved by an average of 2.5ms. Although the object detection accuracy decreased by 6.6% based on mAP 50, the actual output results were not significantly different from the original model. Therefore, the application of FMC and DSC proposed in this paper has been proven effective for object detection in edge computing.

Acknowledgements This results 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. L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *CoRR*, vol. abs/1712.04621, 2017. [Online]. Available: http://arxiv.org/abs/1712.04621
2. C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning - journal of big data," Jul 2019. [Online]. Available: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0citeas
3. K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," *arXiv preprint arXiv:2001.07685*, 2020.
4. Q. Sun, Y. Liu, T. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," 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. 403–412.

5. T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2015.
6. M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–308, September 2009, printed version publication date: June 2010. [Online]. Available: https://www.microsoft.com/en-us/research/publication/the-pascal-visual-object-classes-voc-challenge/
7. D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord, "xview: Objects in context in overhead imagery," 2018.
8. P. Zhu, L. Wen, D. Du, X. Bian, H. Fan, Q. Hu, and H. Ling, "Detection and tracking meet drones challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.
9. S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.
10. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector," in *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007%2F978-3-319-46448-0_2
11. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015. [Online]. Available: https://arxiv.org/abs/1506.02640
12. J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767
13. G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, TaoXie, J. Fang, imyhxy, Lorna, Yifu), C. Wong, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, V. Sonck, tkianai, yxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," Nov. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926
14. Z. Lian and F. Tian, "Deepsi: A sensitive-driven testing samples generation method of whitebox cnn model for edge computing," *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 784–794, 2024.
15. L. Zeng, Q. Liu, S. Shen, and X. Liu, "Improved double deep q network-based task scheduling algorithm in edge computing for makespan optimization," *Tsinghua Science and Technology*, vol. 29, no. 3, pp. 806–817, 2024.
16. G. Wang, Z. P. Bhat, Z. Jiang, Y.-W. Chen, D. Zha, A. C. Reyes, A. Niktash, G. Ulkar, E. Okman, X. Cai, and X. Hu, "Bed: A real-time object detection system for edge devices," 2022.
17. S. Liu, J. Zha, J. Sun, Z. Li, and G. Wang, "Edgeyolo: An edge-real-time object detector," 2023.
18. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
19. T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.