2022 IEEE International Conference on Image Processing (ICIP) | 978-1-6654-9620-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIP46576.2022.9897803

DPNET: DUAL-PATH NETWORK FOR EFFICIENT OBJECT DETECTION WITH LIGHTWEIGHT SELF-ATTENTION

Huimin Shi[1], *Quan Zhou*[1,*], *Yinghao Ni*[1], *Xiaofu Wu*[1], *and Longin Jan Latecki*[2]

[1]National Engineering Research Center of Communications and Networking,
Nanjing University of Posts & Telecommunications, P.R. China.
[2]Department of Computer and Information Sciences, Temple University, Philadelphia, USA.

ABSTRACT

Object detection often costs a considerable amount of computation to get satisfied performance, which is unfriendly to be deployed in edge devices. To address the trade-off between computational cost and detection accuracy, this paper presents a dual path network, named DPNet, for efficient object detection with lightweight self-attention. In backbone, a single input/output lightweight self-attention module (LSAM) is designed to encode global interactions between different positions. LSAM is also extended into a multiple-inputs version in feature pyramid network (FPN), which is employed to capture cross-resolution dependencies in two paths. Extensive experiments on the COCO dataset demonstrate that our method achieves promising detection results. More specifically, DPNet obtains 29.0% AP on COCO test-dev, with only 1.14 GFLOPs and 2.27M model size for a 320 $\times$ 320 image.

Index Terms— Lightweight network, object detection, attention mechanism, convolution neural network

1. INTRODUCTION

Object detection is a challenging task in the area of computer vision, which is widely used in self-driving and robot vision. It aims to detect the minimum bounding boxes that cover objects of interest in input image, and assign an associated semantic label synchronously. Recently, the performance of object detectors has been improved tremendously with the advance of convolution neural networks (CNNs). Typically, object detection can be divided into two-stage and one-stage detectors. Two-stage detectors [1, 2] are always not efficient due to their multi-stage nature. On the contrary, one-stage detectors [3, 4] directly predict object categories and bounding box offsets. Despite of achieving various trade-offs between detection accuracy and running speed, their model size and implementing efficiency are still unacceptable for resource-limited equipments (e.g., drones, robots, and smartphones).

Corresponding author: Quan Zhou, quan.zhou@njupt.edu.cn. This work is partly supported by NSFC (No. 61876093), NSFJS (No. BK20181393), Postgraduate Research & Practice Innovation Program of Jiangsu Province (No. SJCX21_0266), and NSF (No. IIS-1302164).

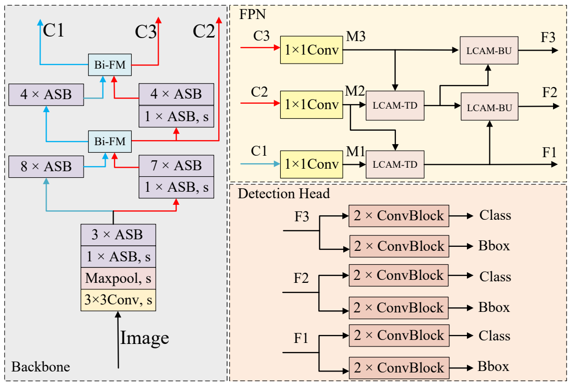

Fig. 1. Overview of DPNet. Our backbone includes a stem and a set of ASBs, resulting in parallel structure of low-resolution path and high-resolution path (denoted by red and blue arrows). Note two paths share stem and first four ASBs. LCAM is designed to enhance cross-scale representations in FPN. Our detection head uses several lightweight ConvBlock units for final prediction. (Best viewed in color)

Although there is a vast number of methods proposed to compress CNNs, such as network quantization [5], network pruning [6], knowledge distillation [7] and compact model design [8], the last category has been dominant for object detection. Derived from [8–10] used for image classification, compact models prefer to directly design lightweight networks in single path architecture. For instance, MobileNet-SSD [8, 9] combines MobileNet with SSD-head. ThunerNet [11] adopts ShuffleNetV2 [10] as backbone by replacing 3 $\times$ 3 depth-wise convolution with 5 $\times$ 5 depth-wise convolution. Pelee [12] employs lightweight backbone with dense structure, reducing output scales of SSD-head to save computation. Tiny-DSOD [13] introduces depth-wise convolutions both in backbone and feature pyramid network (FPN). Tiny-Yolo families [14–16] reduce the number of convolution layers or remove multi-scale outputs in FPN. LightDet [17] employs dilated convolution to enlarge receptive fields in Stem of [10].

On the other hand, due to powerful ability for capturing long-range interactions, self-attention [18–20] starts to show their power for object detection. In [21, 22], some efficient attention mechanisms are embedded in compact networks by

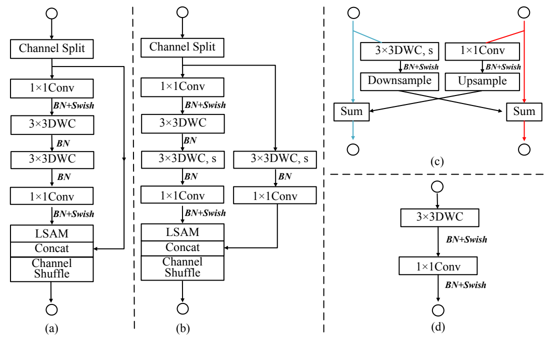

Fig. 2. Overview of the blocks used in backbone and detection head. (a) ASB; and (b) stride version of ASB (s = 2). (c) Bi-FM; (d) ConvBlock; (Best viewed in color)

assigning different weights among feature channels. In spite of achieving remarkable progress, the previous networks inherently suffer from following shortcomings: (1) Single path architecture adopts aggressive downsampling strategies, discarding much fine image details that are useful to locate objects. (2) Although self-attention [18, 19] is powerful to encode global dependency, its computational cost is quadratic to the resolution of input features. On the other hand, some approaches extract global context using global pooling [22,23], yet they ignore the interactions between different spatial positions.

To address these limitations, this paper describes a lightweight parallel path network, called DPNet, for object detection. As shown in Fig. 1, the parallel path architecture leads to dual-resolution subnetwork, extracting high-level semantics in low-resolution path (LRP), and remaining low-level spatial details in high-resolution path (HRP), which are both significant for object detection. Considering the complementary, a bi-direction fusion module (Bi-FM) is constructed to enhance communications between two paths, facilitating the information flow among variable-resolution features. To reduce model size, an Attention-based Shuffle Block (ASB) is designed to build lightweight backbone, in which a single input/output Lightweight Self-Attention Module (LSAM) is employed to capture global interactions. In FPN, LSAM is extended to multiple inputs and single output version, resulting in Lightweight Cross-Attention Module (LCAM) that explores correlations between cross-resolution features. In summary, the contributions of this paper are three-fold:(1) Different from mainstream single-path detector architecture, our DPNet adopts dual-path structure, abstracting both semantic and detail information. (2)To leverage computational efficiency and representation capability, LSAM and LCAM are designed to capture long-range dependency in self and cross-scale terms with acceptable increasing computational costs. (3) We test DPNet on COCO dataset, experimental results demonstrate that DPNet achieves promising trade-off between detection accuracy and implementing efficiency.

2. DPNET

2.1. Backbone

ASB. The backbone of DPNet is mainly constructed by ASB,

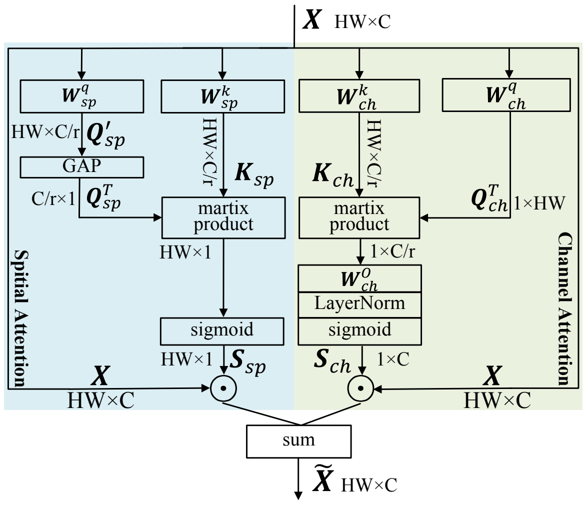

Fig. 3. Overview of LSAM. It includes channel attention and spatial attention. (Best viewed in color)

which is designd to enhance representation ability of global interactions. As depicted in Fig. 2(a), at the beginning of each ASB, input features are split into two lower-dimensional branches, where each one has half channels of the input. One branch remains as identity. Another branch serves as residual branch, following the lightweight bottleneck architecture [9] to extract features. But the channel numbers in this branch remain unchanged, and the standard convolution is replaced by two depth-wise convolution layers. To capture global context, LSAM is designed and its output is concatenated with identity branch. Finally, feature channels are shuffled to enable information communication between two split branches. After the shuffle, the next ASB unit begins. Fig. 2(b) exhibits the stride version of ASB used to reduce feature resolutions, where the 3×3 stride depth-wise convolutions are utilized in identity branch and second convoluiton layer in residual branch.

LSAM. As shown in Fig. 3, given input $\boldsymbol{F} \in \mathbb{R}^{C\times H\times W}$, we first reshape it into a 2D sequence $\boldsymbol{X} \in \mathbb{R}^{HW\times C}$, which is used to compute spatial attention map $\boldsymbol{S}_{sp}$ and channel attention map $\boldsymbol{S}_{ch}$.

In spatial attention, input sequence $\boldsymbol{X}$ is firstly fed into two linear projections $\{\boldsymbol{W}_{sp}^q, \boldsymbol{W}_{sp}^k\} \in \mathbb{R}^{C\times C/r}$ to produce low dimensional embeddings $\{\boldsymbol{Q}'_{sp}, \boldsymbol{K}_{sp}\} \in \mathbb{R}^{HW\times C/r}$, where r is a non-negative scale factor. Thereafter, $\boldsymbol{Q}'_{sp}$ passes a global average pooling $P_g(\cdot)$ to generate $\boldsymbol{Q}_{sp} \in \mathbb{R}^{1\times C/r}$:

$$\boldsymbol{Q}_{sp} = P_g(\boldsymbol{X}\boldsymbol{W}_{sp}^q), \qquad \boldsymbol{K}_{sp} = \boldsymbol{X}\boldsymbol{W}_{sp}^k \tag{1}$$

After that, we perform matrix product between $\boldsymbol{K}_{sp}$ and $\boldsymbol{Q}_{sp}^{\mathrm{T}}$ to calculate spatial correlation map, which undergoes a sigmoid function $\sigma(\cdot)$ to generate spatial attention map $\boldsymbol{S}_{sp} \in \mathbb{R}^{HW\times 1}$:

$$\boldsymbol{S}_{sp} = \sigma(\boldsymbol{K}_{sp}\boldsymbol{Q}_{sp}^{\mathrm{T}}) \tag{2}$$

In channel attention, given two linear projections $\boldsymbol{W}_{ch}^q \in \mathbb{R}^{C\times 1}$ and $\boldsymbol{W}_{ch}^k \in \mathbb{R}^{C\times C/r}$, features $\boldsymbol{Q}_{ch} \in \mathbb{R}^{HW\times 1}$ and $\boldsymbol{K}_{ch} \in \mathbb{R}^{HW\times C/r}$ are similarly generated with $\boldsymbol{Q}_{sp}$ and $\boldsymbol{K}_{sp}$:

$$\boldsymbol{Q}_{ch} = \boldsymbol{X}\boldsymbol{W}_{ch}^q, \qquad \boldsymbol{K}_{ch} = \boldsymbol{X}\boldsymbol{W}_{ch}^k \tag{3}$$

Then, channel correlation map is obtained by matrix product between $\boldsymbol{Q}_{ch}^{\mathrm{T}}$ and $\boldsymbol{K}_{ch}$, which is sequentially fed into a linear projection $\boldsymbol{W}_{ch}^{O} \in \mathbb{R}^{C/r \times C}$, layernorm $LN(\cdot)$, and a sigmoid function to get channel attention map $\boldsymbol{S}_{ch} \in \mathbb{R}^{HW \times C}$:

$$\boldsymbol{S}_{ch} = \sigma(LN(\boldsymbol{Q}_{ch}^{\mathrm{T}} \boldsymbol{K}_{ch} \boldsymbol{W}_{ch}^{O})) \tag{4}$$

Finally, $\boldsymbol{S}_{ch}$ and $\boldsymbol{S}_{sp}$ are used to re-weight input $\boldsymbol{X}$:

$$\widetilde{\boldsymbol{X}} = \boldsymbol{S}_{sp} \odot \boldsymbol{X} \oplus \boldsymbol{S}_{ch} \odot \boldsymbol{X} \tag{5}$$

where $\oplus$ and $\odot$ are element-wise addition and product, respectively. The output sequence $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{HW \times C}$ is reshaped to $\widetilde{\boldsymbol{F}} \in \mathbb{R}^{H \times W \times C}$, and ready for following concatenation.

LSAM leverages computational efficiency and representation ability. From computing perspective, as shown in Fig.3, computing spitial attention map $\boldsymbol{S}_{sp}$ requires $\mathcal{O}(HWC/r)$ operations. The computational complexity of forthcoming reweighting step is $\mathcal{O}(HWC)$. As a result, the computational costs of two steps are $\mathcal{O}(HWC(1 + \frac{1}{r}))$. Recalling we are also required to compute channel attention, thus the total costs are $\mathcal{O}(2HWC(1 + \frac{1}{r}))$. Compared with traditional self-attention [18, 19], the computational costs of our LSAM have been significant reduced since it is linear to HW. From representation perspective, unlike [21, 22] that only utilize global pooling to capture global clues, LSAM encodes long range dependency by computing correlation matrix.

2.2. FPN

Feature pyramid is a fundamental component in state-of-the-art detectors, which mainly aggregates multi-scale features via bilinear interpolation and element-wise addition [24, 25]. This simple fusion strategy ignores dependencies across features with different size, motivating us to extend LSAM into a multi-inputs version. As depicted in Fig. 1, FPN begins at a series of 1×1 convolutions, producing features of $\{$M1, M2, M3$\}$ with equal channel numbers, yet various resolutions. Thus LCAM is adopted to aggerate cross-resolution features in top-down and bottom-up directions (denoted as LCAM-TD and LCAM-BU). LCAM-TD aims to extract high-level semantics for class identification, while LCAM-BU desires to strengthen low-level details for object localization. As they have similar structure, we only elaborate on LCAM-TD, as shown in Fig. 4 (a), for convenience understanding.

LCAM-TD. Let $\boldsymbol{F}_h \in \mathbb{R}^{C \times H_h \times W_h}$ and $\boldsymbol{F}_l \in \mathbb{R}^{C \times H_l \times W_l}$ be high-resolution and low-resolution inputs, we first upsample $\boldsymbol{F}_l$ to $\boldsymbol{X}_l \in \mathbb{R}^{C \times H_h \times W_h}$ that matches resolution of $\boldsymbol{F}_h$, and reshape $\boldsymbol{F}_h$ into a 2D sequence $\boldsymbol{X}_h \in \mathbb{R}^{H_h W_h \times C}$, which are used to compute spatial attention map $\widehat{\boldsymbol{S}}_{sp}$ and channel attention map $\widehat{\boldsymbol{S}}_{ch}$.

In spatial attention, $\boldsymbol{X}_l$ firstly undergoes a 3×3 convolution $\boldsymbol{f}_{3\times3}^{sp}$ to produce a low dimensional embedding $\widehat{\boldsymbol{Q}}_{sp}^{'} \in \mathbb{R}^{C/r \times H_h \times W_h}$, where r is a non-negative scale factor. Thereafter, $\widehat{\boldsymbol{Q}}_{sp}^{'}$ passes global average pooling $P_g(\cdot)$ and reshape operation step-by-step, generating $\widehat{\boldsymbol{Q}}_{sp} \in \mathbb{R}^{C/r \times 1}$. Meanwhile, $\boldsymbol{X}_h$ is fed into a linear projection $\boldsymbol{W}_{sp}^{k} \in \mathbb{R}^{C \times C/r}$ to

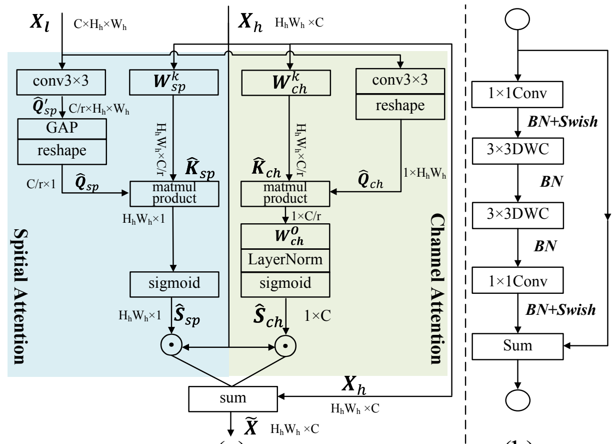

Fig. 4. Overview of the blocks used in FPN. (a) LCAM-TD. (b) Bottleneck. (Best viewed in color)

produce $\widehat{\boldsymbol{K}}_{sp} \in \mathbb{R}^{H_h W_h \times C/r}$:

$$\widehat{\boldsymbol{Q}}_{sp} = P_g(\boldsymbol{f}_{3\times3}^{sp} * \boldsymbol{X}_l), \qquad \widehat{\boldsymbol{K}}_{sp} = \boldsymbol{X}_h \boldsymbol{W}_{sp}^{k} \tag{6}$$

After that, we calculate relation map via matrix product between $\widehat{\boldsymbol{Q}}_{sp}$ and $\widehat{\boldsymbol{K}}_{sp}$, which undergoes a sigmoid function $\sigma(\cdot)$ to generate spatial attention map $\widehat{\boldsymbol{S}}_{sp} \in \mathbb{R}^{H_h W_h \times 1}$:

$$\widehat{\boldsymbol{S}}_{sp} = \sigma(\widehat{\boldsymbol{K}}_{sp} \widehat{\boldsymbol{Q}}_{sp}) \tag{7}$$

In channel attention, given 3×3 convolution $\boldsymbol{f}_{3\times3}^{ch}$ and linear projection $\boldsymbol{W}_{ch}^{k} \in \mathbb{R}^{C \times C/r}$, features $\widehat{\boldsymbol{Q}}_{ch} \in \mathbb{R}^{1 \times H_h W_h}$ and $\widehat{\boldsymbol{K}}_{ch} \in \mathbb{R}^{H_h W_h \times C/r}$ are similarly produced with $\widehat{\boldsymbol{Q}}_{sp}$ and $\widehat{\boldsymbol{K}}_{sp}$ defined in Eqn. (6):

$$\widehat{\boldsymbol{Q}}_{ch} = \boldsymbol{f}_{3\times3}^{ch} * \boldsymbol{X}_l, \qquad \widehat{\boldsymbol{K}}_{ch} = \boldsymbol{X}_h \boldsymbol{W}_{ch}^{k} \tag{8}$$

Similar with Eqn. (4), channel attention map $\widehat{\boldsymbol{S}}_{ch} \in \mathbb{R}^{1 \times C}$ is generated by sequentially performing linear projection $\boldsymbol{W}_{ch}^{O} \in \mathbb{R}^{C/r \times C}$, layernorm $LN(\cdot)$, and sigmoid function $\sigma(\cdot)$:

$$\widehat{\boldsymbol{S}}_{ch} = \sigma(LN(\widehat{\boldsymbol{Q}}_{ch} \widehat{\boldsymbol{K}}_{ch} \boldsymbol{W}_{ch}^{O}) \tag{9}$$

Finally, $\widehat{\boldsymbol{S}}_{ch}$ and $\widehat{\boldsymbol{S}}_{sp}$ are used to re-weight high-resolution input $\boldsymbol{X}_h$, which serves as the residual function to facilitate training LCAM-TD in an end-to-end manner:

$$\widetilde{\boldsymbol{X}} = (\widehat{\boldsymbol{S}}_{sp} \odot \boldsymbol{X}_h \oplus \widehat{\boldsymbol{S}}_{ch} \odot \boldsymbol{X}_h) \oplus \boldsymbol{X}_h \tag{10}$$

The output sequence $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{H_h W_h \times C}$ is reshaped to $\widetilde{\boldsymbol{F}} \in \mathbb{R}^{H_h \times W_h \times C}$, which is fed into a bottleneck module, as shown in Fig. 4(b), and ready for forthcoming feature integration.

LCAM-BU works in a similar way with LCAM-TD except two steps: (1) When computing spatial attention, $\boldsymbol{X}_h$ has to be downsampled to match resolution of $\boldsymbol{X}_l$; (2) $\boldsymbol{F}_l$ has to be reshaped into a 2D sequence with dimension of $H_l W_l \times C$ for feature re-weighting and identity mapping.

2.3. Network Architecture

The overall pipeline of DPNet is shown in Fig. 1. We take the image with size of 320×320 as an input. Concretely, DPNet consists of three components: backbone, FPN and detection head. In backbone, we have dual parallel paths: HRP and LRP. HRP keeps a relatively high resolution (1/8

Table 1. Comparison with the state-of-the-art object detectors in terms of detection accuracy and implementing efficiency on COCO test-dev. AP is COCO standard metric, averagely evaluated at IoU ranged from 0.5 to 0.95 with updated step 0.05.

Method	Year	Backbone	Input Size	FLOPs (G)	Params (M)	AP (%)	AP_{50} (%)	AP_{75} (%)	FPS
Tiny-YOLOV3 [15]	Arxiv2018	Tiny-DarkNet	416×416	2.78	8.7	16.0	33.1	–	368
MobileNet-SSD [9]	Arxiv2017	MobileNet	300×300	1.2	6.8	19.3	–	–	59.3
Tiny-YOLOV4 [16]	CVPR2021	Tiny-CSPDarkNet	416×416	3.45	6.1	21.7	40.2	–	**371**
MobileNetV2-SSDLite [8]	CVPR2018	MobileNetV2	320×320	0.8	4.3	22.1	–	–	–
MobileNet-SSDLite [8]	CVPR2018	MobileNet	320×320	1.3	–	22.2	–	–	–
Pelee [12]	NeurIPS2018	PeleeNet	304×304	1.29	6.0	22.4	38.3	22.9	125
Tiny-DSOD [13]	BMVC2018	DDB-Net	300×300	1.12	**1.15**	23.2	40.4	22.8	105
LightDet [17]	ICASSP 2020	LightNet	320×320	**0.5**	–	24.0	42.7	24.5	250
MobileDets [23]	CVPR2021	IBN+Fused+Tucker	320×320	1.43	4.85	26.9	–	–	–
ThunderNet [11]	ICCV2019	SNet535	320×320	1.3	–	28.1	**46.2**	29.6	214
Ours	-	DPNet	320×320	1.14	2.27	**29.0**	45.2	**30.4**	178

Table 2. We validate the effectiveness of proposed modules on COCO val set step-by-step with input resolution 320×320.

LRP	HRP	Bi-FM	LSAM	LCAM	Params(M)	Flops(G)	AP(%)
✓					1.77	0.83	24.0
✓	✓				1.91	1.05	25.3
✓	✓	✓			2.11	1.11	26.6
✓	✓	✓	✓		2.39	1.23	28.4
✓	✓	✓	✓	✓	2.27	1.14	29.0

of image size), while LRP employs multi-step downsampling operations to obtain low-resolution features (1/32 of image size). As shown in Fig. 2(c), Bi-FM is designed to enhance cross-resolution feature communication. Our backbone has multi-outputs {C1, C2, C3}, whose shapes are {$40 \times 40 \times 128$, $20 \times 20 \times 256$, $10 \times 10 \times 512$}. FPN takes {C1, C2, C3} as inputs, and produces intermediate features {M1, M2, M3} with equal feature channels of 128. Then, they are fed into a series of LCAMs in top-down and bottom-up manner, producing outputs of FPN {F1, F2, F3}. These fused cross-resolution features pass two lightweight ConvBlocks, as shown in Fig. 2(d), to predict bounding boxes and categorical labels.

3. EXPERIMENTS

3.1. Experimental Settings

Dataset. Experiments are implemented on the widely-used challenging detection benchmark MS COCO 2017 [26], which contains 118K training, 5K validation and 20K test-dev images. Our model is trained on the train set. A system-level comparison is reported on test-dev.

Implementation Details. DPNet is trained from scratch with a minibatch of 76 images on a single RTX 2080Ti GPU. We adopt stochastic gradient descent (SGD) for 300 epochs with 5 epochs warm up. Cosine learning strategy is used with an initial learning rate, weight decay, and momentum set as $1.5 \times e^{-2}$, $5 \times e^{-4}$, and 9×10^{-1}, respectively. We adopt half-precision (FP16) to reduce GPU memory usage. Loss functions and other settings follow [27]. We follow SSD [3] to augment training data, which is useful to boost performance.

3.2. Evaluation Results on MS COCO

Tab. 1 reports comparison results with selected state-of-the-art lightweight detectors, demonstrating that DPNet achieves best trade-off in terms of detection accuracy and implementation efficiency. DPNet achieves 29.0% AP on COCO test-dev, with only 2.27M model size and 1.14 GFLOPs. Among all baselines, DPNet is nearly 2× model size than Tiny-DSOD [13], but improves detection performance with 5.8% AP. LightDet [17], another lightweight detectors, has approximately half GFLOPs of DPNet, but delivers poor detection results of 5.0% drop in terms of AP. Although ThunderNet [11] outperforms our DPNet in terms of AP$_{50}$, we improve AP and AP$_{75}$ by large margins of 0.9% and 0.8%, respectively, indicating that DPNet performs better in object localization. At last, the running speed of DPNet is 178 FPS with a 320×320 input image, which is beneficial to some real-time applications in timely fashion.

3.3. Ablation Studies

To investigate the effectiveness of proposed modules, we conduct ablation studies on COCO val set, where HRP, Bi-FM and LSAM are added step-by-step. Tab. 2 reports the results. It is observed that LRP only leads to 24.0% AP without considering fine spatial information. The introduction of HRP brings 1.3% AP improvement with only 0.14M model size and 0.22 GFLOPs increase. To facilitate communication between two paths, Bi-FM obtains 1.3% performance gain in terms of AP. As a core unit of our DPNet backbone, LSAM boosts performance by 1.8% AP with slight increase of GFLOPs (0.12) and number of parameters (0.28M). Finally, instead of using heavy FPN [4] that involves standard 3×3 convolution, our LCAM reduces 0.12M model size and 0.09 GFLOPs, while acquires 0.6% AP score improvement.

4. CONCLUSION REMARKS AND FUTURE WORK

This paper has described a DPNet for lightweight object detection, which extracts high-level semantics and low-level details using dual-resolution representations. In backbone, LSAM is designed in ASB to capture global context, while remains lightweight architecture. We further extend LSAM to LCAM in FPN that encodes global interactions between cross-resolution features. The experimental results on COCO benchmark show that DPNet achieves state-of-the-art trade-off in terms of detection accuracy and implementing efficiency. In the future, we are interested in transferring DPNet to instance segmentation [28] and keypoint estimation [29].

5. REFERENCES

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *NeurIPS*, vol. 28, pp. 91–99, 2015.

[2] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *ICCV*, pp. 2961–2969, 2017.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, pp. 21–37, 2016.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, pp. 779–788, 2016.

[5] Z. Wang, J. Lu, Z. Wu, and J. Zhou, "Learning efficient binarized object detectors with information compression," *IEEE T-PAMI*, 2021.

[6] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," *NIPs*, vol. 28, 2015.

[7] X. Dai, Z. Jiang, Z. Wu, Y. Bao, Z. Wang, S. Liu, and E. Zhou, "General instance distillation for object detection," in *CVPR*, pp. 7842–7851, 2021.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, pp. 4510–4520, 2018.

[9] 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.

[10] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *ECCV*, pp. 116–131, 2018.

[11] Z. Qin, Z. Li, Z. Zhang, Y. Bao, G. Yu, Y. Peng, and J. Sun, "Thundernet: Towards real-time generic object detection on mobile devices," in *ICCV*, pp. 6718–6727, 2019.

[12] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," in *NeurIPS*, pp. 1967–1976, 2018.

[13] J. L. Yuxi Li, Jianguo Li and W. Lin, "Tiny-DSOD: Lightweight object detection for resource-restricted usage," in *BMVC*, pp. 6718–6727, 2018.

[14] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *CVPR*, pp. 7263–7271, 2017.

[15] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[16] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in *CVPR*, pp. 13024–13033, 2021.

[17] Q. Tang, J. Li, Z. Shi, and Y. Hu, "Lightdet: A lightweight and accurate object detection network," in *ICASSP*, pp. 2243–2247, 2020.

[18] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *CVPR*, pp. 7794–7803, 2018.

[19] J. Fu, J. Liu, H. Tian, Y. Li, Y. Bao, Z. Fang, and H. Lu, "Dual attention network for scene segmentation," in *CVPR*, pp. 3146–3154, 2019.

[20] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *ICCV*, 2021.

[21] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, pp. 7132–7141, 2018.

[22] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi, "Gather-excite: Exploiting feature context in convolutional neural networks," in *NeurIPS*, vol. 31, 2018.

[23] Y. Xiong, H. Liu, S. Gupta, B. Akin, G. Bender, Y. Wang, P.-J. Kindermans, M. Tan, V. Singh, and B. Chen, "Mobiledets: Searching for object detection architectures for mobile accelerators," in *CVPR*, pp. 3825–3834, 2021.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *CVPR*, pp. 2117–2125, 2017.

[25] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *CVPR*, pp. 8759–8768, 2018.

[26] 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," in *ECCV*, pp. 740–755, 2014.

[27] X. Li, W. Wang, L. Wu, S. Chen, X. Hu, J. Li, J. Tang, and J. Yang, "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection," *NeurIPS*, vol. 33, pp. 21002–21012, 2020.

[28] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "Solo: Segmenting objects by locations," in *ECCV*, pp. 649–665, 2020.

[29] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *CVPR*, pp. 5693–5703, 2019.