Backbone Based Feature Enhancement for Object Detection

Haoqin Ji[1,2], Weizeng Lu[1,2], and Linlin Shen[1,2](✉)

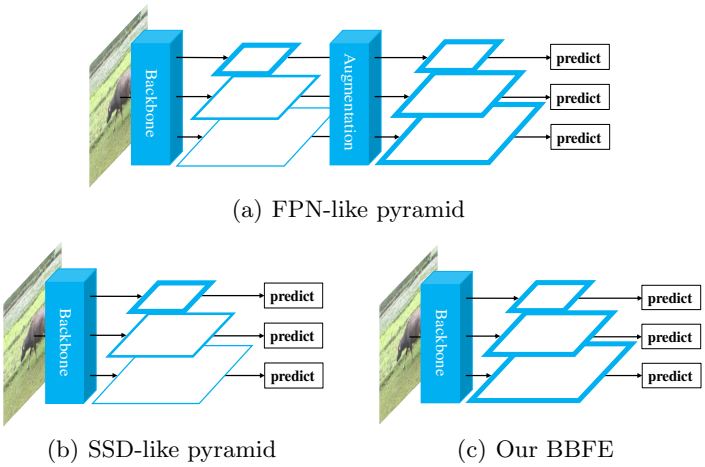
[1] Computer Vision Institute, School of Computer Science and Software Engineering, Shenzhen University, China
[2] Shenzhen Institute of Artificial Intelligence and Robotics for Society, China
{jihaoqin2019, luweizeng2018}@email.szu.edu.cn, llshen@szu.edu.cn

Abstract. FPN (Feature Pyramid Networks) and many of its variants have been widely used in state of the art object detectors and made remarkable progress in detection performance. However, almost all the architectures of feature pyramid are manually designed, which requires ad hoc design and prior knowledge. Meanwhile, existing methods focus on exploring more appropriate connections to generate features with strong semantics features from inherent pyramidal hierarchy of deep ConvNets (Convolutional Networks). In this paper, we propose a simple but effective approach, named BBFE (Backbone Based Feature Enhancement), to directly enhance the semantics of shallow features from backbone ConvNets. The proposed BBFE consists of two components: reusing backbone weight and personalized feature enhancement. We also proposed a fast version of BBFE, named Fast-BBFE, to achieve better trade-off between efficiency and accuracy. Without bells and whistles, our BBFE improves different baseline methods (both anchor-based and anchor-free) by a large margin (∼2.0 points higher AP) on COCO, surpassing common feature pyramid networks including FPN and PANet.

Keywords: Feature enhancement · Object detection.

1 Introduction

As one of the most fundamental and challenging tasks in computer vision, object detection aims to accurately detect the objects of predefined categories in digital images. Benefited from the development of deep ConvNets (Convolutional Networks) [1] and GPUs computing power in recent years, modern object detectors (e.g., SSD [2], YOLO [3], RetinaNet [4], Faster R-CNN [5], Cascade R-CNN [6]) have achieved impressive progress. However, object detection still faces many challenges, e.g. there usually exists objects with various scales in the same picture. To solve this problem, a traditional idea is to use an image pyramid [7] to build a feature pyramid and then detect objects of different scales in different feature maps, which is effective but brings huge computing cost. As shown in Fig. 1(b), SSD (Single Shot Detector) [2] is one of the first attempts to combine the predictions from pyramidal feature hierarchy directly to handle



(a) FPN-like pyramid

(b) SSD-like pyramid

(c) Our BBFE

Fig. 1. (a) Designing additional augmentation to enhance detection capability of original features. (b) Using pyramidal feature hierarchy that have semantics from low to high levels. (c) Building a strong pyramid from backbone directly. In this figure, thicker outlines denote semantically stronger features.

objects of various sizes, but the performance is limited due to the limited semantics available in shallow level feature maps. After that, FPN (Feature Pyramid Network) [8] proposes to create a feature pyramid with strong semantics at all scales by developing an efficient top-down pathway with lateral connections after the bottom-up pathway, which shows significant improvements with small extra cost.

The success of FPN attracts wide attention from the community. After that, a series of subsequent works, such as FPR [9], PANet [10], and M2Det [11], propose to further improve the pyramid architectures like FPN from different aspects to obtain enriched features. As shown in Fig. 1(a), Most of the existing works related to pyramidal feature fusion can be summarized into two steps: (1) select one or more feature maps from multiple levels of deep ConvNets and denote them as $\{C_2, C_3, \dots\}$; (2) carefully design the multi-scale features fusion module and augment $\{C_2, C_3, \dots\}$ to $\{P_2, P_3, \dots\}$ that contain richer semantic information. Finally, these high-level features are applied to the subsequent detection process. Although these works promote detection accuracy remarkably, designing a suitable feature fusion scheme is of great challenge, which requires rich experience and prior knowledge.

The feature hierarchy computed by backbone has an inherent multi-scale, pyramidal shape, which we call SSD-like pyramid. In SSD-like pyramid, while the deepest feature map has the strongest semantic features, the shallower feature maps have lower-level features. What if we continue to apply the subsequent layers of ConvNets to the shallow feature maps of the SSD-like pyramid? Inspired by that, we proposed a new method, named BBFE (Backbone Based Feature

Enhancement), to build strong feature pyramids. As shown in Fig. 1(c), the basic motivation of BBFE is to: (1) avoid using hand-crafted feature fusion architecture and (2) directly produce a feature pyramid that has strong semantics at all scales from backbone. To achieve this goal, we first generate an SSD-like pyramid from a deep ConvNet, as most other detectors do. Next, we employ the RBW (reusing backbone weight) module to reuse the original convolutional layers of backbone ConvNet and enhance the semantics of shallow-layer features, without introducing extra parameters. Then we attach a very simple PFE (personalized feature enhancement) block on each semantically stronger feature, to further boost the detection performance with marginal extra cost. After that, we proposed a faster version of BBFE, named Fast-BBFE, to achieve better trade-off between accuracy and speed.

In our experiments, we compared the standard FPN with our BBFE using different types of detection algorithms, and demonstrate competitiveness of the proposed method. The main contributions of this work can be summarized as:

- Propose a RBW module to get semantically strong features directly from the standard backbone, without additional hand-crafted architectures;
- Propose a more powerful BBFE approach consisting of RBW and a light-weight detection neck PFE, which can be generalized to all detectors and replace FPN for better detection performance;
- Without using any tricks, our approach achieves a consistent improvement ($\sim$2.0 points higher AP) on both anchor-based and anchor-free detectors.

2 Related Work

Benefited from the success of deep ConvNets [1] and the great increase in computing power, the CNN-based object detectors have greatly advanced the performance of hand-crafted features [12, 13]. In order to efficiently detect objects of various scales simultaneously in a single image and further push the upper bound of detection accuracy, deep object detectors usually adopt multiple feature layers. Using image pyramid [7] to construct feature pyramid is a reasonable solution, but it is time-consuming because of the large computation burden to independently generate features on each image scale. To better address this issue, lots of works have been proposed.

Applying Multi-layer Features. Reusing feature maps from different layers computed in the forward propagation of ConvNet is a useful way to deal with the problem of scales. SSD [2] directly utilizes several original feature maps of backbone layers to improve detection performance. SDP+CRC [14] uses features in multiple layers to detect objects of different scales by using the proposed scale-dependent pooling and cascaded rejection classifiers. MSCNN [15] applies deconvolutional layers on multiple feature maps of a CNN to increase their resolutions, and later these refined feature maps are used to make predictions. RFBNet [16] proposes a multi-branch convolution block similar to the Inception block, and combine them with the dilated convolution to further enhance the discriminability and robustness of features.

Fusing Multi-layer Features. Recent works exploit lateral or skip connections to fuse information between different layers to produce combined features. DSOD [17] follows the SSD framework and fuses multi-scale prediction responses with an elaborated dense structure. FPN [8] passes semantic information from deep layers to shallow layers by a concise top-down pathway, which further shows the power of feature fusion. Furthermore, a series of improved FPN are proposed. PANet [10] adds another bottom-up path with lateral connections on the basis of FPN to shorten the information path. FPR [9] adaptively concatenates multiple layers extracted from backbone network and then spread semantics to all scales through a more complex module. Libra R-CNN [18] proposes a BFP (Balanced Feature Pyramid) to aggregates multiple features and refine the integrated feature with the non-local module [19], then scatter it to all scales. NAS-FPN [20] adopts NAS (Neural Architecture Search) [21] to find a new feature pyramid architecture, which consists of free connections to fuse features of different scales. Other representative methods, such as DSSD (Deconvolutional Single Shot Detector) [22], TDM (Top Down Modulation) [23], STDN (Scale Transfer Detection Network) [24], RefineDet [25] and M2Det [11], also made impressive progress in multi-layers fusion.

Most of the existing studies, except NAS-FPN, manually design the connection between backbone features and pyramidal features. Considering that it might be ad hoc to manually design architectures for fusing features across scale, our work focuses on building a semantic feature pyramid directly from the backbone, which is a simple and near parameter-free method that only reuses the original convolutional layers.

3 Method

The overall pipeline of BBFE is shown in Fig. 2. Taking a single-scale image of an arbitrary size as input, our goal is to directly build a high-level semantic feature pyramid by reusing original layers. All the components and details will be introduced in this section.

3.1 Reusing Backbone Weight

As shown in Fig. 2, the dotted box represents the proposed RBW (reusing backbone weight) module, which is based on the backbone ConvNet's pyramidal feature hierarchy $\{C_2, C_3, C_4, C_5\}$. $\{C_2, C_3, C_4, C_5\}$ are the output of different convolutional blocks and have semantics from low to high levels. The deepest layer C_5 has the strongest semantics and can be used for subsequent detection without further improvement. For other layers, we continue to apply the later convolutional blocks to shallower features to enrich their semantics. For example, C_3 is the output of conv2, so we continue to apply conv3 and conv4 to C_3 to enrich its semantic information. When reusing a convolutional block, we need to change the down-sampling operation in order to keep the spatial size of output

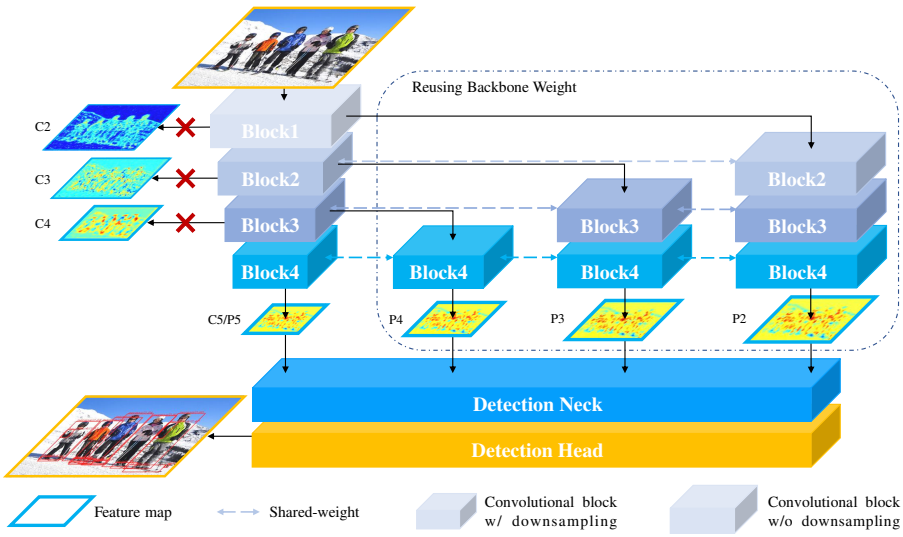


Fig. 2. Overview of the proposed BBFE. $\{C_2, C_3, C_4, C_5\}$ denotes the original pyramidal feature hierarchy of backbone ConvNets, C_5 and P_5 are identical. Feature maps with red crosses are not required and will not be generated in our approach. We firstly enhance the middle-level features $\{C_2, C_3, C_4\}$ into semantically strong features $\{P_2, P_3, P_4\}$ by reusing the backbone weight (no extra parameters) while maintaining their original resolutions. Then $\{P_2, P_3, P_4, P_5\}$ are further enriched by detection neck (e.g. the PFE block) before the subsequent detection.

and input layer consistent. To be specific, if the down sampling operation is related to the stride of the convolution kernel, we will reset the stride and padding value. If the downsampling operation is associated with the pooling layer, we simply skip that layer.

We strengthen multi-scale feature presentations by reusing the original convolution blocks, with the removal of down-sampling operation. Compared to previous studies, our method can obtain semantic features without requirement of complex structures for feature fusion and feature enhancement. The proposed RBW does not require any additional parameters and can be easily generalized to other detectors. It's a parameter-free approach for feature pyramid enhancement.

3.2 Personalized Feature Enhancement

Object detectors leverage feature pyramid to detect small objects from high resolution images and detect large objects from low resolution images. Because many convolution kernels are shared for feature extraction, RBW module may generate similar semantic information for feature maps generated by different branches, which limit the representation ability cross scales. To alleviate the problem, we

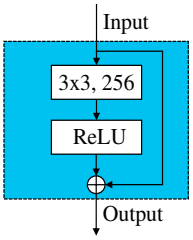


Fig. 3. The overview of the personalized feature enhancement block.

introduce the PFE (personalized feature enhancement) block shown in Fig. 3 and separately attach it on each layer to make features more discriminative. To avoid complex manual design, we simply adopt a residual block consisting of a 3×3 convolution and ReLU function. Inspired by ResNet [26], our hypothesis is that the shortcut enables input features to retain their original representation if extra convolution destroys the learned semantic information. Based on RBW, the PFE block can further improve the detection accuracy with marginal extra cost.

In addition, the detection neck (PFE block) can be replaced by other existing components (e.g. Non-Local [19]) for better accuracy, which may introduce too much calculations. In this paper, simplicity is central to our design.

3.3 Fast-BBFE

The combination of RBW module and PFE block forms a simple and effective approach named BBFE (backbone based feature enhancement) for enhancing

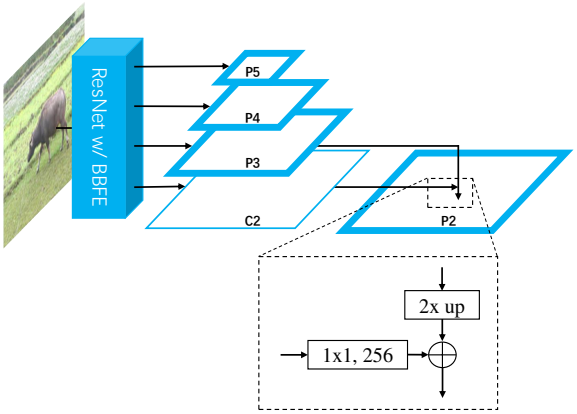


Fig. 4. The overview of the Fast-BBFE: Using BBFE to obtain semantic features except C_2. P_2 is generated by merging C_2 and P_3 after simple dimension transformation.

detection ability of each feature map. The speed of BBFE is inversely proportional to the scale of the feature map. Backbone ConvNet computes a feature hierarchy consisting of feature maps at several scales with a scaling step of 2, and the largest feature map will cost most of the time due to the removal of down-sampling.

To address this issue, we apply BBFE to the original features except the biggest one, which is built through FPN-way separately. We denote the output of each residual blocks of ResNets as $\{C_2, C_3, C_4, C_5\}$, and note that they have strides of $\{4, 8, 16, 32\}$ pixels with respect to the input image. We firstly utilize BBFE to enhance $\{C_3, C_4, C_5\}$ to $\{P_3, P_4, P_5\}$, and then generate P_2 by merging C_2 and P_3, as shown in Fig. 4. We up-sample the spatial resolution of P_3 by a factor of 2 using nearest neighbor up-sampling, and use a 1×1 convolutional layer to reduce the channel dimensions of C_2, then merge the up-sampled map with C_2 by element-wise addition to produce P_2. Unlike FPN, we use the PFE block to reduce the aliasing effect of up-sampling. We call BBFE with this module Fast-BBFE, which achieves better trade-off between accuracy and efficiency. As Table 2 shows, RetinaNet [4] using Fast-BBFE surpasses FPN based RetinaNet in both accuracy and speed.

4 Object Detection

State of the art object detectors available in literature can be mainly categorized into anchor-based and anchor-free approaches. Our method is a generic solution for building strong semantic feature pyramids based on backbone ConvNets. In this section, we generalize BBFE to different types of object detectors. To demonstrate the effectiveness and simplicity of BBFE and Fast-BBFE, we make minimal and reasonable modifications to these algorithms to adapt them to our methods.

4.1 Anchor-based Detector

RetinaNet [4] and Faster R-CNN [5] are widely used one-stage and two-stage methods. As shown in Fig. 5(a), RetinaNet uses feature pyramid levels P_3 to P_7, where P_3 to P_5 are computed from the output of the corresponding residual stage (C_3 through C_5) of ResNets [26] using FPN. P_6 is computed via a 3×3 stride-2 convolution on P_5, and P_7 is obtained by applying ReLU followed by a 3×3 stride-2 convolution on P_6. Since we did not use FPN, in order to get the same number of feature layers as RetinaNet, we simply append two 3×3 stride-2 convolution layers on ResNets to generate C_6 and C_7, which are then enhanced to P_6 and P_7 by BBFE. Fig. 5(b) shows the architecture of RetinaNet with BBFE.

Faster R-CNN based on FPN (for short we still call it Faster R-CNN) uses feature pyramid levels P_2 to P_6, where P_2 to P_5 are the output of ResNets corresponding to C_2 to C_5, and P_6 is generated by applying max-pooling on P_6.

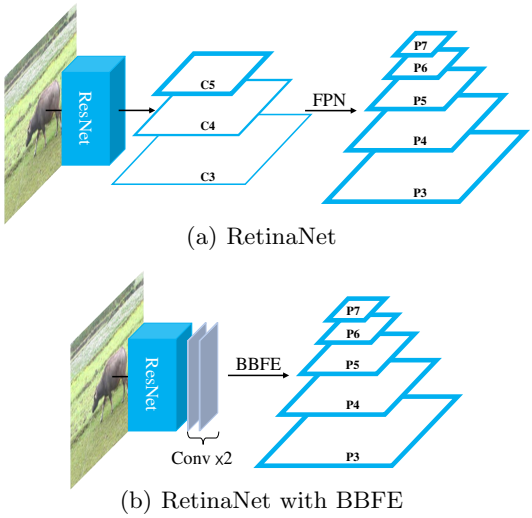


Fig. 5. (a) Original RetinaNet that uses FPN by default. (b) Replacing FPN with BBFE and appending two extra 3x3 conv layers on backbone to generate the same number of feature maps as FPN.

Analogous to the RetinaNet combined with BBFE, we firstly add a 3x3 convolutional layer to Faster R-CNN backbone and generate the feature hierarchy $\{C_2, C_3, C_4, C_5, C_6\}$, where C_6 is the output of the extra conv layer, then use BBFE to directly obtain feature pyramid $\{P_2, P_3, P_4, P_5, P_6\}$.

4.2 Anchor-free Detector

Anchor-free detectors can be divided into center-based and keypoint-based methods. For example, FCOS [27] proposes a novel centerness score and predicts object bounding box with four distances. RepPoints [28] represents objects as a set of sample points and learns to automatically arrange themselves in a particular manner. Although the algorithm details are different, both FCOS and RepPoints use common backbone networks and detection necks to extract and enhance features. Similar to RetinaNet, we replace FPN with BBFE in FCOS and RepPoints using the approach shown in Fig. 5.

5 Experiments

5.1 Datasets and Evaluation Metrics

We conduct all experiments on the challenging COCO dataset [29], which consists of 115k images for training (*train-2017*) and 5k images for validation (*val-2017*). We use the *train-2017* split for training and perform all ablation studies on the *val-2017* subset, then report our main results on the *test dev* split (20k

Table 1. Comparison with the state-of-the-art methods on COCO *test-dev*. The symbol "*" means our implementation. The number in [ ] denotes the relative improvement. We exclude all the training and testing tricks in our experiments for fair comparison. All the settings are the same as the default settings provided in the MMDetection.

Method	Backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv3 [33]	DarkNet-59	33.0	57.9	34.4	18.3	35.4	41.9
SSD513 [2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [22]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RefineDet512 [25]	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4
Mask R-CNN [34]	ResNet-101-FPN	38.2	60.3	41.7	20.1	41.1	50.2
Mask R-CNN [34]	ResNeXt-101-FPN	39.8	62.3	43.4	22.1	43.2	51.2
Libra R-CNN [18]	ResNet-101-FPN	40.3	61.3	43.9	22.9	43.1	51.0
ExtremeNet [35]	Hourglass-104	40.2	55.5	43.2	20.4	43.2	53.1
CornerNet [36]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
Faster R-CNN*	ResNet-50-FPN	36.5	58.6	39.2	21.5	39.4	44.7
FCOS*	ResNet-50-FPN	37.1	56.5	39.6	20.5	39.8	46.7
RepPoints*	ResNet-50-FPN	38.6	59.7	41.5	22.7	41.8	47.2
RetinaNet*	ResNet-50-FPN	35.8	56.0	38.3	20.0	39.0	43.9
RetinaNet*	ResNet-101-FPN	38.0	58.7	40.8	21.3	41.6	47.5
RetinaNet*	ResNeXt-101-FPN	40.2	61.3	43.5	23.1	43.9	50.9
Faster R-CNN (ours)	ResNet-50-Fast-BBFE	38.1[+**1.6**]	59.9	41.5	23.2	40.9	47.4
FCOS (ours)	ResNet-50-Fast-BBFE	38.3[+**1.2**]	57.9	40.9	22.0	40.9	48.4
RepPoints (ours)	ResNet-50-Fast-BBFE	39.5[+**0.9**]	60.8	42.5	23.6	42.7	48.6
RetinaNet (ours)	ResNet-50- Fast-BBFE	36.9[+**1.1**]	57.5	39.7	21.4	40.2	45.5
RetinaNet (ours)	ResNet-101- Fast-BBFE	38.9[+**0.9**]	59.6	42.0	22.3	42.4	49.0
RetinaNet (ours)	ResNeXt-101- Fast-BBFE	41.1[+**0.9**]	62.4	44.5	23.9	44.8	51.7
Faster R-CNN (ours)	ResNet-50-BBFE	38.6[+**2.1**]	60.4	41.7	23.7	41.3	47.6
FCOS (ours)	ResNet-50-BBFE	39.4[+**2.3**]	58.9	42.2	24.1	41.7	48.9
RepPoints (ours)	ResNet-50-BBFE	40.3[+**1.7**]	61.4	43.5	25.2	43.2	48.8
RetinaNet (ours)	ResNet-50-BBFE	37.8[+**2.0**]	58.5	40.8	22.6	40.8	46.7
RetinaNet (ours)	ResNet-101-BBFE	40.0[+**2.0**]	60.7	43.3	24.0	43.1	49.9
RetinaNet (ours)	ResNeXt-101-BBFE	41.6[+**1.4**]	62.8	44.8	25.0	45.0	52.4

images with disclosed labels) by uploading our detection results to the evaluation server. All the experiment results are reported using standard COCO-style AP (Average Precision) metrics, including AP, AP_{50}, AP_{75}, AP_S, AP_M, AP_L.

5.2 Implementation Details

All the experiments are implemented based on PyTorch [30] and MMDetection [31] for fair comparisons. For the main results, we select four representative detectors as our baseline, and use ResNet backbone initialized with the weights pretrained on ImageNet [32]. The input images are resized to have their shorter side being 800 and their longer side less or equal to 1333. We use 4 GPUs (2 images per GPU) to train detectors with SGD (Stochastic Gradient Descent) for 12 epochs, which is commonly referred as 1x training schedule. According to the linear scaling rule, the initial learning rate was set to 0.005 and decreased by 0.1 after epoch 8 and 11. Weight decay and momentum are set as 0.0001 and 0.9, respectively. All other hyper-parameters follow the settings in MMDetection.

5.3 Main Results

Without loss of generality, we verify the effectiveness of BBFE and Fast-BBFE on different types of detectors and compare them with other state-of-the-art detectors on *test dev* split of COCO benchmark. The results are reported in Table 1. Many researches adopt longer training schedule and scale jitters as well as multi-scale/flip testing to make their detectors achieve better results. For a fair comparison, we exclude all the training and testing tricks in all experiments. We firstly reimplement the baseline methods equipped with FPN, which generally perform better than that reported in their papers. Then we make minimal changes to the original object detectors to fit our BBFE and Fast-BBFE introduced in Section 4.

As Table 1 shows, by replacing FPN with Fast-BBFE and BBFE, Faster R-CNN based on ResNet-50 achieves 38.1 AP and 38.6 AP, respectively, which is 1.6 points and 2.1 points higher than ResNet50-FPN based Faster R-CNN. As for one-stage detector, The AP of RetinaNet with BBFE using different backbones is improved by 1.4 $\sim$ 2.0 points. As for anchor-free detectors, BBFE improves the AP of FCOS and RepPoints by 2.2 and 1.6 points, respectively, when using ResNet50 as backbone. The improvement of RepPoints is relatively small, because the original RepPoints uses Group Normalization [37] in standard FPN to boost performance. Without bells and whistles, The proposed BBFE bring significant improvements to various backbones and detectors, which demonstrates the robustness and generalization ability of our method.

5.4 Ablation Studies

Overall Ablation Studies. For a better understanding to our approach, we further conduct a series of ablation studies and report results in Table 2. We gradually add RBW (reusing backbone weight), PFE (personalized feature enhancement) and the fast version of the combination of the previous two modules on ResNet-50-FPN RetinaNet600 baseline. The first row of Table 2 shows the result of the original RetinaNet600, which use FPN by default. We replace the FPN architecture in RetinaNet with our methods and show the corresponding results in rows 2 through 6.

Table 2. Ablation studies on COCO *test-dev*.

w/ RBW?	w/ PFE?	Fast-BBFE?	FPS	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
			13.3	34.4	54.0	36.7	17.8	38.3	47.0
✓			10.8	36.1	56.3	38.8	19.7	40.0	48.2
	✓		13.4	34.4	54.1	36.8	18.0	38.4	46.7
✓		✓	14.6	35.0	54.7	37.2	18.4	39.2	47.1
✓	✓		10.7	**36.4**	**56.4**	**39.1**	**20.4**	**40.5**	**48.5**
✓	✓	✓	**14.6**	35.4	55.4	38.0	18.6	39.6	47.7

Reusing Backbone Weight. Reusing backbone weight achieves 1.7 points higher box AP than the ResNet-50-FPN RetinaNet600 baseline, demonstrating the power of this simple approach, which does not introduce any additional parameters or utilize any FPN-like feature fusion methods.

Personalized Feature Enhancement. Personalized feature enhancement further improves the box AP from 36.1 to 36.4, where most of the improvements are from AP_S, i.e. 0.7 points increase. Row 5 of Table 2 indicates that the PFE block can boost performance with marginal extra cost (FPS reduced from 10.8 to 10.7).

Fast-BBFE. Based on BBFE, Fast-BBFE achieves a better trade-off in accuracy and efficiency. The last row in Table 2 shows that Fast-BBFE achieves better precision and increasing speed over the baseline, which can even compete with ResNet-50-FPN RetinaNet800 (35.5 AP and 12.6 FPS).

Table 3. Comparisons with other feature pyramidal architectures on COCO *test-dev*.

Architecture	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
FPN [8]	35.5	55.6	38.1	21.2	39.3	45.9
BFP [18]	35.7	56.2	38.0	20.8	40.2	46.2
PANet [10]	35.9	55.9	38.4	20.9	40.1	46.3
FPN + BFP	36.4	56.9	38.6	21.2	40.4	48.1
BBFE (ours)	**37.3**	**57.6**	**40.0**	**22.9**	**41.2**	**47.6**
BBFE + FPN	36.8[−0.5]	57.8	39.4	23.4	40.4	46.3
BBFE + PANet	38.2[+0.9]	58.5	**41.2**	23.7	**42.4**	**49.4**
BBFE + BFP	**38.3**[**+1.0**]	**59.3**	41.0	**24.7**	42.1	47.8

Comparisons with Other Feature Pyramid. To further demonstrate the effectiveness of our method, we compared in Table 3 with common hand-crafted feature pyramidal networks including FPN, BFP, and PANet. Our experiments are performed on ResNet-50 RetinaNet with the image scale of [1333, 800]. NAS-FPN is not involved because its experimental conditions in MMDetection are quite different from other algorithms. As Table 3 shows, our BBFE yields 0.9 $\sim$ 1.8 points higher AP over other FPNs on the same baseline. In particular, compared with the method of FPN combined with BFP in row 4, BBFE has a more balanced increase in AP_S (+1.7 points), AP_M (+1.9 points) and AP_L (+1.7 points), which further demonstrates that our method consistently enhances the feature map of all scales.

Our BBFE is able to achieve 37.3 AP on COCO dataset, which is 1.8 points higher than ResNet-50-FPN RetinaNet. Furthermore, we tried to combine BBFE with other feature pyramid structures to achieve further improvements and the corresponding results are shown in the last three rows of Table 3. BBFE increases the AP of PANet and BFP by 0.9 and 1.0 points, respectively. However, using standard FPN after BBFE gets worse performance. As FPN focus on passing the semantic information down to the shallow layers one by one to enhance low-

level features, it is not suitable for features already containing strong semantic information, which is the case of BBFE.

6 Discussion

Why can BBFE Improve Detection Performance? We take ResNet-50 Faster R-CNN (without FPN) for example. The feature activations output by all stages of ResNet-50 are denoted as $\{C_2, C_3, C_4, C_5\}$, which have semantics from shallow to deep layers. Inspired by CAM [38], we visualize the features to better understand their properties. As shown in Fig. 6, the redder area means richer semantic information. Intuitively, low-level features focus on the edge information, while high-level features focus on the semantic information of the object. The shallow layers are not suitable for detection due to the weak semantics. BBFE use the same convolutional layers on feature maps of different scales, which greatly alleviates this problem. The comparison of C_2 and P_2, C_3 and P_3 as well as C_4 and P_4 in Fig. 6 shows that BBFE can add semantic information into low level features and retain the original edge feature at the same time, so as to reasonably improve the detection capability.

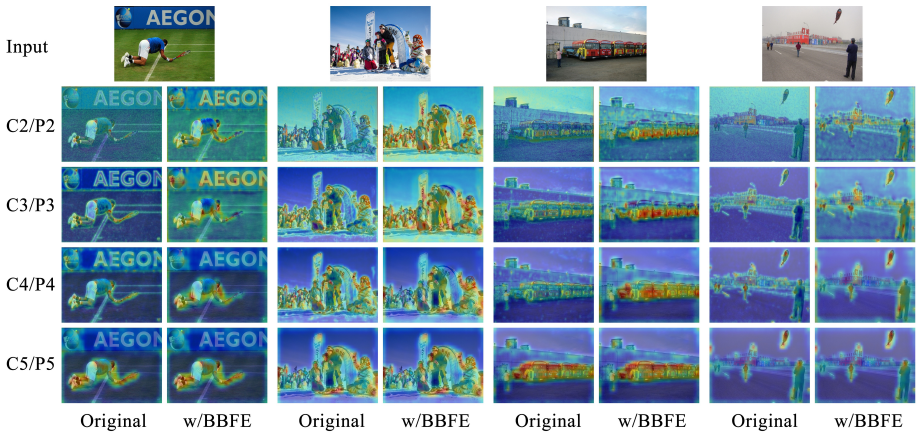


Fig. 6. Heatmap visualization. From top to bottom: the original image, feature maps from different depths of ConvNets. The odd columns represent the original features, and the even columns represent features enhanced by BBFE.

Runtime Analysis. ResNet-50-FPN can run at 12.6 FPS on 800 × 1333 input images, which outperforms ResNet-50-Fast-BBFE (11.7 FPS) . However, as listed in Table 2, ResNet-50-Fast-BBFE runs at 14.6 FPS while ResNet-50-FPN only operates at 13.3 FPS on 600 × 1000 input images. Although the proposed BBFE obtains feature maps in parallel, the inference time of BBFE is limited on the slowest branch. Increasing the input resolution cause heavy computations,

due to the size of largest feature maps. Conventional FPN runs slower than Fast-BBFE on smaller images as it needs to wait for the backbone to generate pyramidal feature hierarchy. However, it is less sensitive to the image resolutions, due to the down-sampling layers. All the runtimes are tested on a single Tesla V100 GPU.

7 Conclusion

In this paper, we propose a near parameter-free approach, named BBFE, to directly build a semantic feature pyramid from detection backbone without additional feature fusions. Compared with existing FPNs, our BBFE is simple and requires no hand-crafted effort by exploring the potential of backbone ConvNets. The experiments on COCO dataset demonstrate that our method can produce significant improvements upon mainstream detectors without any training or testing tricks. We also proposed Fast-BBFE to achieve better trade-off between accuracy and efficiency. Moreover, our approach can be combined with other FPNs to further improve the performance of object detection.

References

1. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. (2012) 1097–1105
2. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision, Springer (2016) 21–37
3. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 779–788
4. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. (2017) 2980–2988
5. Ren, S., He, K., Girshick, R., Jian, S.: Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **39** (2015)
6. Cai, Z., Vasconcelos, N.: Cascade r-cnn: Delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 6154–6162
7. Adelson, E.H., Anderson, C.H., Bergen, J.R., Burt, P.J., Ogden, J.M.: Pyramid methods in image processing. RCA engineer **29** (1984) 33–41
8. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 2117–2125
9. Kong, T., Sun, F., Tan, C., Liu, H., Huang, W.: Deep feature pyramid reconfiguration for object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 169–185

10. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2018) 8759–8768
11. Zhao, Q., Sheng, T., Wang, Y., Tang, Z., Chen, Y., Cai, L., Ling, H.: M2det: A single-shot object detector based on multi-level feature pyramid network. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 33. (2019) 9259–9266
12. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Volume 1., IEEE (2005) 886–893
13. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60** (2004) 91–110
14. Yang, F., Choi, W., Lin, Y.: Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2129–2137
15. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection. In: European conference on computer vision, Springer (2016) 354–370
16. Liu, S., Huang, D., et al.: Receptive field block net for accurate and fast object detection. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 385–400
17. Shen, Z., Liu, Z., Li, J., Jiang, Y.G., Chen, Y., Xue, X.: Dsod: Learning deeply supervised object detectors from scratch. In: Proceedings of the IEEE international conference on computer vision. (2017) 1919–1927
18. Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., Lin, D.: Libra r-cnn: Towards balanced learning for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 821–830
19. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 7794–7803
20. Ghiasi, G., Lin, T.Y., Le, Q.V.: Nas-fpn: Learning scalable feature pyramid architecture for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 7036–7045
21. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)
22. Fu, C.Y., Liu, W., Ranga, A., Tyagi, A., Berg, A.C.: Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 (2017)
23. Shrivastava, A., Sukthankar, R., Malik, J., Gupta, A.: Beyond skip connections: Top-down modulation for object detection. arXiv preprint arXiv:1612.06851 (2016)
24. Zhou, P., Ni, B., Geng, C., Hu, J., Xu, Y.: Scale-transferrable object detection. In: proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 528–537
25. Zhang, S., Wen, L., Bian, X., Lei, Z., Li, S.Z.: Single-shot refinement neural network for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 4203–4212
26. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778

27. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9627–9636
28. Yang, Z., Liu, S., Hu, H., Wang, L., Lin, S.: Reppoints: Point set representation for object detection. In: Proceedings of the IEEE International Conference on Computer Vision. (2019) 9657–9666
29. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755
30. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems. (2019) 8026–8037
31. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al.: Mmdetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
32. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115** (2015) 211–252
33. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
34. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. (2017) 2961–2969
35. Zhou, X., Zhuo, J., Krahenbuhl, P.: Bottom-up object detection by grouping extreme and center points. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2019) 850–859
36. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 734–750
37. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). (2018) 3–19
38. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2921–2929