Abstract—In order to tackle the issue of multi-scale object detection, recent detectors usually adopt hierarchical feature pyramids which are generated by naive combinations of top-down features and lateral features. Considering the limited effective receptive fields of the methods for top-down features augmentation, the generated regions are only associated with the fixed areas of the coarser features. Meanwhile, noisy features are introduced by irrelevant regions inevitably since the finer features in relation to rigid coarser regions. Thus, the pyramidal features with strong semantics are difficult to be obtained via simply enlarging the top-down features. In this paper, we present the Aggregated Residual Dilation based Feature Pyramid Network (ARDFPN) to exploit the inherent correlation of regions in feature pyramid. The network is designed by stacking a building block that aggregates a set of dilated convolutions with the same topology. We show that carefully adding additional transformation stages into feature pyramid enables a potential way for further multi-scale feature generation. As an intuitive extension of Feature Pyramid Network (FPN), we conduct an exhaustive study to evaluate the model performance by replacing FPN with the proposed ARDFPN in both object detection and instance segmentation tasks. With Residual network in Faster R-CNN and Mask R-CNN framework, ARDFPN outperforms the prevalent detection module — FPN on the challenging COCO dataset without bells and whistles. In particular, ARDFPN exhibits a superior performance, especially for the small and middle objects.

Index Terms—Object Detection, Neural Networks, Feature Extraction

I. INTRODUCTION

In the last decades, Convolutional Neural Networks (CNNs) based methods [1], [2], [3], [4], [5] have proven to be effective in object detection task [6]. With deep convolutional networks (ConvNets), detectors can generate highly semantical feature maps in deep layers (Fig. 1(a)). Nevertheless, such deep feature maps lack accurate localization information and clues of small objects by sub-sampling layers. Although shallow layers contain adequate finer details of objects, semantically strong feature maps can not be extracted in them for detection. Thus, developing a scale-invariant detector is still a fundamental challenge in computer vision.

X. Zhao, Y. Zhang, S. Chang, and Z. Feng are with the Key Laboratory of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing, P. R. China, 100876, email: {zhaoxiaotong, zhangyf, changshuo, fengzy}@bupt.edu.cn.

P. Zhang is with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, P. R. China, 100876, email: pzhang@bupt.edu.cn.

W. Li is with the Department of Electrical Engineering, Northern Illinois University, Dekalb IL, USA 60115, email: weili@ieee.org.

Corresponding author: Wei Li (weili@ieee.org).

Research on scale-invariant feature generation is experiencing a transition from “hand-crafted engineering” [7], [8], [9], [10] to “network engineering” [2], [11], [12], [13], [14] during the past twenty years. The hand-crafted methods [7], [8], [9], [10], which are also referred to as the non-deep-learning based methods, generate multi-scale feature maps by heavily using the featurized image pyramid. For instance, detectors like SIFT [7], HOG [8], and DPM [9] require the image pyramid of densely sampled scales to produce an appropriate scale space for detection. In order to generate the finely-sampled feature pyramid with fewer computations, the well-known detection scheme — Fast Feature Pyramid [10] is presented with Aggregated Channel Features (ACF). Here, the multi-scale feature maps can be estimated at octave-spaced scale intervals without detection rates reduction. However, such hand-crafted detectors [7], [8], [9], [10] can not produce robust feature maps persistently over a long range of scales. Moreover, the representation abilities of features designed by hand [7], [8], [9] stay in a shallow-level where the semantics are relatively weak for detection with the increasing variety of objects.

In contrast with traditional featurized image pyramids which the pyramidal levels built upon hand-crafted features, deep network based feature pyramids [11], [12], [13], [14] is one of the promising solution for multi-scale detection (Fig. 1(b)). These structures produce feature maps at different scales via combinations of top-down pathways and lateral connections, in which all pyramidal layers have properly semantical representation. By scanning the feature pyramid, objects can be detected at a long range of scales through all-level semantically strong feature maps without information loss. But still, the expected semantically strong pyramidal feature maps with sufficient information are usually generated through a naive transformation for coarse feature maps [11], [12]. More specifically, the coarse feature maps are simply enlarged by interpolation methods or transposed convolutions, while the merged fine feature maps are connected through convolutional mappings. For instance, as one of the first deep ConvNets that attempting to adopt hierarchical feature pyramid for object detection, the Feature Pyramid Network (FPN) [11] upsamples the low-resolution feature maps using nearest neighbor algorithm by a factor of 2. And the final high-resolution feature maps are realized by element-wise additions of enlarged feature maps and corresponding lateral feature maps. Without considering the region correlation in coarser feature maps, such methods overlook the procedure of pyramidal feature generation, which result in sub-optimal detection performance, especially for
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2941892, IEEE Access

segmentation tasks by using the information of hierarchical features, we empirically demonstrate that ARDFPN outperforms the original FPN module in both object detection and instance segmentation. Regardless of the backbone convolutional architecture, a well-designed implementation of semantical connections, a well-designed implementation of semantical feature generation can further enhance the pyramidal layers.

In this paper, we present a fully convolutional network — Aggregated Residual Dilation based Feature Pyramid Network (ARDFPN), which leverages the region correlation in each pyramidal feature generation block based on the split-transform-merge philosophy. As shown in Fig. 1(c), instead of simply using interpolation or transposed convolution, the generation block of the feature pyramid conducts a set of dilated convolutional transformation in a transposed residual manner, whose outputs are aggregated by element-wise summation. Through multiple transformation paths, each generation block exploits the region correlation spatially with different dilation rates. As the dilation rate increases, the new regions are generated from a wide range of contexts. The effective receptive fields of the top-down pathway in the feature pyramid are increased by the instruction of the aggregated dilation blocks. Additionally, with larger transposed convolution kernels, the generated pyramidal feature maps can involve more related coarser regions, which further enhances the semantical representation. Regardless of the backbone convolutional architectures, we empirically demonstrate that ARDFPN outperforms the original FPN module in both object detection and instance segmentation tasks by using the information of hierarchical feature maps more efficiently. With the help of the aggregated dilation module and transposed residual learning, ARDFPN is able to facilitate the semantic generation of feature pyramid, which improves the final detection performance.

The main contributions of this work can be summarized as follows:

1) To the best of our knowledge, we are the first to give a detailed analysis about the impact of effective receptive field in the procedure of the feature pyramid generation. Here, we discuss the effect differences of dilated convolutions between feature extraction and feature generation for the instances of different scales (small, middle, and large).

2) We develop a novel ARDFPN to deal with the multi-scale issue for object detection. As far as we know, the proposed ARDFPN is the first architecture that utilizes a matrix of dilated convolutions to generate scale-invariant pyramidal features with stronger semantics. And we present the controlled experiments to exploit the effectiveness of receptive fields for pyramidal feature generation by using different dilation rates choice in ARDFPN.

3) We design a transposed residual learning framework to ease the training of the generation blocks in the feature pyramid, thus producing the multi-scale feature maps with better representation ability. Experiment results demonstrate the proposed transposed residual learning is critical for the hierarchical feature generation.

4) We introduce the influence of the different transposed convolution kernels for pyramidal feature generation. Through appropriate transposed convolution kernel, the generative transformation can bring more contextual information and complement the “holes” caused by dilated convolutions.

5) We validate the effectiveness of the proposed ARDFPN on the well-known COCO benchmark with exhaustive ablation studies. Compared to the prevalent detection

---

Fig. 1 (a) Detectors that utilizes single scale feature maps, which is computational friendly. (b) Feature pyramid methods that generate multi-scale features by naive combinations of top-down pathways and lateral connections. Here the FPN [11] is used for comparison. (c) The proposed ARDFPN restores scale-aware feature maps by Aggregated Dilation Blocks (ADB) with different effective receptive fields in transposed residual manner. The red and green lines indicate plain convolutions and transposed convolutions (or interpolations), respectively. The dashed lines represent that there are other consecutive layers which are not shown in the figure for simplicity.

---

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.
II. RELATED WORKS

A. Object Detection

In the pioneering work of Girshick et al. [1], the two-stage detector — R-CNN was first presented by integrating segmentation algorithms [15], [16] into AlexNet [17], which classifies each candidate location over 2000 generated region proposals. Rather than utilizing a large number of segmented images, SPPNet [18] proposes a spatial pyramid pooling layer to re-use the feature maps for region proposals generation regardless of the input image size. However, such methods [1], [18] still depend on segmentation performance to obtain hypothetical object locations. In [2], Ren et al. presented a feature map sharing network — Faster R-CNN, which adopts Region Proposal Network (RPN) to predict both segmented locations and objectness scores. To equilibrate the distinction between translation-variance of object detection and translation-invariance of image classification, Dai et al. [4] introduced R-FCN with Position Sensitive Region of Interest pooling (PSRoI) to replace the original RoI pooling in Faster R-CNN.

In order to achieve real-time object detection, the one-stage detectors [3], [19], [20] are proposed in recent years, which conduct bounding boxes regression and classification over densely predetermined regular object locations. Redmon et al. [3] applied YOLO to detect objects on several grid cells, that each cell can be considered as a predefined region proposal. To improve the accuracy of one-stage detectors, SSD [19] was presented with multi-scale detection. SSD is a fully convolutional network, which leverages multiple feature layers to classify objects over dense default boxes. Since the foreground-background class imbalance happens when training one-stage detectors, Lin et al. [20] proposed the Focal Loss to address this issue, as well a dense detector named RetinaNet for evaluation.

Rather than designing a scale-invariant detector, Singh et al. [21] adopted a scale-aware training scheme named Scale Normalization for Image Pyramids (SNIP). With image pyramids, SNIP solves the domain-shift problem by selecting the gradients of RoIs that suitable for the scale of the pre-trained model during the back-propagation procedure. Instead of using the whole image for training, SNIPER [5] was introduced to process images as multi-scale context regions that around ground-truth instances (referred to as chips). SNIPER can get rid of the dependence on high-resolution images in the training procedure and saves the GPUs usage significantly.

B. Hierarchical Feature Maps

The utilization of hierarchical feature maps can be traced back to the “hand-crafted engineering” methods [7], [8], [9], [10], [22], [23], [24]. In the previous work of Lazebnik et al. [22], the Spatial Pyramid Representation (SPR) was presented to encode the global and local spatial information into one feature, which exhibited remarkable performance in both object and scene recognition tasks. By repeatedly subsampling the image and computing the hierarchical feature maps, an orthogonal spatial pyramid match kernel is used as the image representation in SPR. In order to improve the partition strategy of SPR, which is designed by hand and demands expensive computational cost, Harada et al. [23] introduced an adaptive feature representation scheme named Discriminative Spatial Pyramid Representation (DSPR). DSPR forms the discriminative features by an automatically-selected weighted summation of hierarchical features. Although the issue of multi-scale detection is alleviated to some extent by using aggregated features, such detectors [22], [23] can not guarantee the performance in the scenarios that the scales of objects are varied dramatically. Therefore, the noted methods like SIFT [7], HOG [8], and DPM [9] still need densely sampled image pyramid to generate multiple hierarchical feature maps to satisfy a variety of scales. Nevertheless, generating a dense set of scales requires a large computational price, which had become the bottleneck of the hand-crafted detectors. Hence, an effective detection scheme named Fast Feature Pyramid [10] was proposed to yield faster detection speed with comparable detection performance. Based on the feature scaling power law [25], the proposed scheme approximates finely sampled pyramid by extrapolation from coarsely sampled octave layers, which is inexpensive compared to computing multi-scale feature maps directly. Lately, Hong et al. [24] presented an integrated hierarchical method for multispectral palmprint recognition. To develop the features that can better represent the scale and orientation for textures, the method [24] fuses the rough feature — Block Dominant Orientation Code (BDOC) and the fine feature — Block-based Histogram of Oriented Gradient (BHOG) from different bands to increase the recognition accuracy.

Considering the weak semantics of the hand-crafted features, recent deep learning based object detectors start to pay attention on multi-scale detection by incorporating hierarchical feature maps into networks [11], [12], [13], [14], [26], [27], [28], [29]. By leveraging feature maps from different layers within CNN, Cai et al. [26] proposed MS-CNN to combine the complementary scale-specific detectors into a unified detection framework. Inspired by the pathway of the human visual system, Shrivastava et al. [27] introduced TDM network, which merges finer details of objects from lower layers with high-level features through top-down modulation. To achieve independent predictions on multi semantic layers, Lin et al. [11] presented FPN to generate pyramidal feature maps, which combines semantically strong feature maps with localization oriented feature maps via top-down and lateral pathways. To further introduce additional contextual information, DSSD [12] was applied to integrate transposed convolution into SSD.
Instead of the original single prediction layer, DSSD uses multiple prediction modules for the final detection. Due to the difficulty of effective feature fusion in SSD, Li et al. [28] proposed FSSD with a lightweight feature fusion module. FSSD generates a new feature pyramid by the concatenated multi-scale features. On account of the huge computational cost of multi-scale feature maps generation, Liu et al. [29] explicitly explore a recurrent scale approximation to predicate feature maps of various scales. Other than simply merging multiple feature maps from different network levels, Zhou et al. [14] introduced STDN embedded with scale-transfer layers to exploit the scale consistency for object detection. By using the DenseNet [30] as the backbone network, STDN adopts scale-transfer layers and pooling layers to generate high-resolution feature maps for detecting small and large objects, respectively. In order to realize comparable accuracy with two-stage detectors while maintaining the efficiency of one-stage detectors, Zhang et al [13] presented RefineDet which transfers the refined anchors to predict the categorizations and locations of objects in a single-shot manner.

C. Dilated Convolution

To acquire the global prediction with sufficient semantical representations, networks [17], [30], [31], [32] for image classification task [33] reduce the resolution of input images by successive pooling and downsampling layers. However, tasks like dense prediction and object detection call for higher-resolution outputs for spatially recognize and locate the object instances. Recently, approaches [34], [35], [36], [37] that involve dilated convolutions (also denoted as atrous convolutions) [38] show promising performance in dense prediction and object detection tasks, which aim to extract feature maps in a larger receptive field without resolution losing. Through dilated convolution, the receptive field can be enlarged via adding “holes” into the original convolution kernel with no additional parameter cost. Yu et al. [34] developed a dilated-convolution-based module that combined multi-scale contextual information for semantical segmentation. To further improve the performance of segmenting objects at multiple scales, DeepLabs [35], [36] are presented with modules that employ dilated convolution in cascade or in parallel. By setting different dilation rates in the kernel of dilated convolution, DeepLabs can capture multi-scale contextual information. For the purpose of designing an accurate detector with real-time processing speed, Liu et al. [37] proposed the RFB module to enhancing the features generated by lightweight network based on the structure of human population Receptive Fields (pRFs). The RFB module makes use of an aggregated spatial array of the receptive field, which is built by multi-branch convolution kernels with different dilated convolutional layers, to simulate the properties of pRFs. For the sake of enlarging the receptive field while keeping the spatial size of feature maps, Li et al [39] introduced a specific backbone network named DetNet by adopting dilated convolutions, which boosts the detection accuracy of large objects significantly. The methods mentioned above all use dilated convolutions in their backbone network. For all we know, the ARDFPN is the first attempt to exploit the ability of multi-branch dilated convolutions in feature pyramid generation.

III. MODEL DESIGN

In this section, we give a comprehensive introduction of Aggregated Residual Dilation based Feature Pyramid Network (ARDFPN) for object detection. The proposed ARDFPN consists of multi-level Aggregated Dilation Blocks (ADBs) based on transposed residual learning. Moreover, an elaborate transposed convolution kernel is studied to further improve the detection performance. The overall framework of ARDFPN is illustrated in Fig. 2.

A. Aggregated Dilation Block

In order to compensate for the absence of contextual information in finer pyramidal feature generation procedure, ARDFPN is constructed by adding multiple ADBs in the top-down pathway. The goal of ADB is to restore the higher-resolution feature maps of multi-scale instances through aug-
Aggregated Dilation Block

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2941892, IEEE Access

Contrast to only applying the neighbor regions in low-resolution, more coarser regions are interleaved with each other. In contrast to the effective receptive fields in each pyramidal level. Hence, we employ a multi-branch architecture in ADB through parallel dilated convolutions with different dilation rates to accommodate the receptive fields for instances of different scales. In each branch, low-resolution feature maps can be further enhanced by cascaded dilated or non-dilated convolutional layers of $3 \times 3$ kernels with stride 1. Since each convolutional layer is followed by an activation layer [40], [41], [42], integrating such non-linear layers into ADB brings more discriminate representations for the feature transformation. Furthermore, due to the learnable weighted combinations of coarser regions in convolutional layers, the feature noises caused by unrelated regions can be eliminated to a certain degree. Then aggregated feature maps are obtained by element-wise summation of different branches. By using dilated convolutions, the pyramidal feature maps are generated maintaining the spatial resolution with enlarged receptive fields. The structure of ADB is illustrated in Fig. 3(b).

Here, we use “dilation rate matrix” $\Theta \in \mathbb{R}^{W \times D}$ to describe the architecture of ADB, where $W$ and $D$ are the width and depth of ADB, respectively. Thus the dilation rate of the particular layer in ADB is represented as $\Theta_{ij}$, where $i = 1, 2, ..., W$ and $j = 1, 2, ..., D$ stand for the index of width and depth, respectively. Formally, the aggregated dilated convolution is presented as:

$$F(x) = \sum_{i=1}^{W} T_i(x|\Theta_{i1}, \Theta_{i2}, ..., \Theta_{iD}),$$

where $T_i(x)$ denotes the cascaded transformation. As an essential factor for feature map transformation in ARDFPN, we will discuss the specific parameters of “dilation rate matrix” in Sec. IV-C and Sec. IV-D.

As shown in Fig. 3, by increasing the number of branches and inserting more “holes” into the plain convolutional layers, more coarser regions are interleaved with each other. In contrast to only applying the neighbor regions in low-resolution feature maps, which are conducted in previous methods [11], [12], [27], the utilization of ADBs can restore the original location and detail information of multi-scale instances by introducing various dilated convolution in parallel, which captures a long-range of relevant contextual information for pyramidal feature generation. Formally, for each pyramidal level, we give the receptive field of each layer in ADB as

$$r_{i0} = 1,$$

$$\tilde{k}_{ij} = k_{ij} + (k_{ij} - 1) \times (\Theta_{ij} - 1),$$

$$r_{ij} = r_{i,j-1} \times \tilde{k}_{ij} - (\tilde{k}_{ij} - 1) \times (r_{i,j-1} - \prod_{k=1}^{j-1} s_k),$$

where $k_{ij}$ denotes the kernel size and $s_k$ represent the stride of each layer.

Here, we deem that the effects of leveraging dilated convolutions are different between feature extraction and feature generation. While for feature extraction procedure, which is implemented in the backbone of the network, the dilated convolutions are usually beneficial to the recognition of large instances from the increasing receptive fields [39]. Nevertheless, in the feature generation procedure, the usage of dilated convolutions are tended to improve the detection performance of middle or small objects by enlarging the accessible context.

### B. Transposed Residual Learning

In the feature pyramid, the top-down pathway that transforming the coarser feature map to its corresponding finer version can be considered as a generative procedure. Inspired by the residual learning [32], such generative transformation should be much easier to learn the residual mapping with reference to the low-resolution input. Instead of directly learning the underlying mapping for feature transformation, we explicitly let the ADB adopt a residual function by using transposed residual learning. As illustrated in Fig. 2, the pyramidal feature map is obtained by merging of three parts, i.e., top-down residual pathway, top-down shortcut, and lateral connection. Concretely, the top-down residual pathway is used to learn the residual contents between the optimal pyramidal feature map and the upsampled feature map, which are difficult to be optimized by unreferenced mapping. While the top-down shortcut is proposed for upsampling the identity contents of coarser feature map with learnable weights. And the generated feature map is further refined through lateral connection for better semantical representation. We show the performance of residual transformation via the ablation study in Sec. IV-G.

Formally, we define the transposed residual learning as

$$\bar{x} = \max(x, 0) + k_1 \min(x, 0),$$

$$\tilde{y} = \Phi(F(\bar{x})) + \Phi(\bar{x}),$$

$$y = \max(\tilde{y}, 0) + k_2 \min(\tilde{y}, 0) + z,$$

where $x$ and $y$ are the input and output of the pyramidal level, while $z$ represents the lateral feature map. The function $\Phi$ stands for the transposed convolution and the learned residual mapping is denoted as $\Phi(F(\bar{x}))$. Since the input feature map of each pyramidal level is usually generated by convolutional

---

Fig. 3 The comparison between the plain convolution and the proposed Aggregated Dilation Block for finer feature generation.
we utilize the channel-shared PReLU \cite{41} as the activation mapping without activation \cite{11}, we first pre-activate the input kernels for finer feature map generation. Here \( k, p, s \) denote kernel size, padding, and stride, respectively.

C. Larger Transposed Convolution Kernel

By replacing the interpolation methods with transposed convolutions for enlarging coarser feature maps, the procedure of pyramidal feature generation can be benefited from the adjustable hyper-parameters and learnable weights. In this paper, we define the transposed convolution kernel through a tuple of hyper-parameter, i.e., (kernel size, padding, stride). In feature pyramid, the kernel size is responsible for the number of related regions of coarser feature map for the finer generation procedure, while the padding is used to adjust the size of the pyramidal feature map along with the kernel size. And the stride is regarded as the amplification factor of each pyramidal level.

The Fig. 4 shows a comparison diagram of different transposed convolution kernels for finer feature map generation. Here \( k, p, s \) denote kernel size, padding, and stride, respectively.

D. Loss Function

For training ARDFPN with the Faster R-CNN framework in an end-to-end manner, the overall loss function is a summation of the RPN loss of all the feature pyramid levels and the Fast R-CNN loss:

\[
L = \sum_{l} L_{RPN_{l}} + L_{FastRCNN},
\]

where \( l \) is the index of the feature pyramid level. Being the same with FPN, we define anchors of RPN in the feature pyramid from level 2 to 6, where they have scales from \( 32^2 \) to \( 512^2 \), respectively. And the aspect ratios of anchors are set to \{1:2, 1:1, 2:1\} at each level. Formally, the RPN loss of each feature pyramid level is defined as follows \cite{2}:

\[
L_{RPN}(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_{i} L_{cls}(p_i, p^*_i)
+ \frac{\lambda}{N_{reg}} \sum_{i} [p^*_i = 1] L_{reg}(t_i, t^*_i),
\]

where \( i \) is the index of the anchor in each feature pyramid level \cite{11}. The predicted probability and ground-truth label are denoted as \( p_i \) and \( p^*_i \), respectively. The ground-truth label \( p^*_i \) is set to 1 if the anchor is the foreground, and 0 for the background. Here \( t_i \) is the parameterized 4-tuple coordinates \cite{1} of the predicted bounding box, and \( t^*_i \) is the ground-truth parameterized coordinates of the corresponding anchor. The Verson bracket \([\cdot]\) indicates to 1 when the inner discriminant is true and 0 otherwise. The classification loss \( L_{cls} \) is computed by the sigmoid function with the binary cross-entropy loss \cite{43} and the regression loss \( L_{reg} \) is the smooth \( L_1 \) loss \cite{44}. The two losses are normalized by \( N_{cls} \) and \( N_{reg} \), where \( N_{cls} \) is fixed to the predefined RPN batch size (e.g. = 256 in \cite{2}, \cite{11}) and \( N_{reg} \) is the number of the training anchors, which is equal or lesser than the predefined number. A parameter \( \lambda \) is used to balance these two terms, which is set to 1 by default.

Following \cite{44}, the Fast R-CNN loss is defined as:

\[
L_{FastRCNN}(\{p_j\}, \{t_j\}) = \frac{1}{N_{Rois}} \sum_{j} (L_{cls^*}(p_j, p^*_j)
+ \eta [p^*_j \geq 1] L_{reg}(t_j, t^*_j)),
\]

where \( j \) is the index of the RoI, which is generated according to the region proposal collect-distribute scheme used in \cite{11}. Similar to the definitions mentioned above, \( p_j \) and \( p^*_j \) are the predicted probability and ground-truth label of the RoI, while \( t_j \) and \( t^*_j \) are the parameterized 4-tuple coordinates of the predicted bounding box and the regression target. Here, the classification loss \( L_{cls^*} \) is computed by the softmax function with the multinomial cross-entropy loss \cite{44}. The Fast R-CNN loss \( L_{FastRCNN} \) is normalized by \( N_{Rois} \), which is represented for the number of the training RoIs. The balancing parameter \( \eta \) is set to 1 by default.

As mentioned in \cite{43}, for training ARDFPN with the Mask R-CNN framework in an end-to-end manner, the overall loss \( L^* \) is constructed by simply replacing the Fast R-CNN loss \( L_{FastRCNN} \) with the Mask R-CNN loss \( L_{MaskRCNN} \) in the Faster R-CNN loss:

\[
L^* = \sum_{l} L_{RPN_{l}} + L_{MaskRCNN},
\]
in which
\[ L_{MaskRCNN}\{\{p_j\}, \{t_j\}, \{m_j\}\} = \frac{1}{N_{RoIs}} \sum_j (L_{cls}(p_j, p_j^*) + \eta[p_j^* \geq 1]L_{reg}(t_j, t_j^*)) + \frac{\mu}{N_{fg}} \sum_j [p_j^* \geq 1]L_{mask}(m_j, m_j^*), \]

where \( m_j \) and \( m_j^* \) are the predicted mask and ground-truth mask for each RoI. The mask loss \( L_{mask} \) is defined by the average binary cross-entropy loss. For each RoI with the corresponding ground-truth category \( c \), \( L_{mask} \) is only computed on the \( c \)-th mask. Different from \( L_{cls} \) and \( L_{reg} \), which are normalized by the number of the RoIs, \( L_{mask} \) is normalized by \( N_{fg} \), which is the number of the foreground of the RoIs. The balancing parameter of the mask loss \( \mu \) is set to 1 by default as well.

In addition, we give the regression loss \( L_{reg} \) with a complete definition of the smooth \( L_1 \) loss:
\[ L_{reg}(t, t^*) = \sum_{k \in \{x, y, w, h\}} \text{smooth}_{L_1}(t_k - t_k^*), \]

in which
\[ \text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < \beta, \\ |x| - 0.5\beta, & \text{otherwise}, \end{cases} \]

where \( \beta \) is the transition point between \( L_2 \) loss and \( L_1 \) loss. It should be noted that \( \beta \) is set to \( \frac{1}{\sqrt{2}} \) for \( L_{FPN} \), and 1 for \( L_{FastRCNN} \) and \( L_{MaskRCNN} \).

IV. EXPERIMENTS

In this section, the proposed ARDFPN is evaluated by the well-known MS COCO dataset [6], which is one of the most commonly used benchmarks for object detection. For object detection task, there are 80 categories and more than 100k images for evaluation. As in previous works [11, 39, 43], we train ARDFPN using the union of 80k training images and 35k validation images, and report the detection results on the minival set of 5k images. We use the standard MS COCO evaluation metric in all experiments, including AP\(_{\text{IoU}=0.50}^{}\), AP\(_{\text{IoU}=0.75}^{}\), AP (averaged over Intersection-over-Union (IoU) values from 0.50 to 0.95), and AP\(_{\text{small}}^{}\), AP\(_{\text{medium}}^{}\), AP\(_{\text{large}}^{}\) to assess the performance of ARDFPN with different IoU thresholds and scales. Here the metrics are also shown as AP\(_{50}^{}\), AP\(_{75}^{}\), AP, AP\(_{50}^{}\), AP\(_{75}^{}\), AP\(_{s}^{}\) for simplicity.

We first introduce the implementation details and training settings in Sec. IV-A. Then we compare the bounding boxes detection results of ARDFPN with the original C4 [43] and FPN [11] by different backbones and detection frameworks in Sec. IV-B. We discuss the appropriate depth and width of ARDFPN in Sec. IV-C. And in Sec. IV-D, we conduct experiments with various combinations of dilatation rates. Sec. IV-E and Sec. IV-F give discussions about transposed convolution kernel and activation function, respectively. An ablation study is performed in Sec. IV-G. In order to evaluate the recall performance of ARDFPN, we present the average recall experiment in Sec. IV-H. Finally, the proposed ARDFPN is adopted for instance segmentation to exhibit the generality in Sec. IV-I.

A. Implementation Details

Following the common practice [11, 39, 43], the backbone network of ARDFPN is pre-trained by ImageNet [33] first and then the whole network is finetuned on the MS COCO dataset. By default, we train ARDFPN using the Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a weight decay of 0.0001 for 90k iterations in total. The learning rate is initialized to 0.02 and is divided by 10 after 60k and 80k iterations. The warmup scheme [45] for learning rate is adopted in the first few epochs of training. The batch for each iteration involves 16 images with 256 anchors and 512 RoIs per image. Moreover, the weights of ARDFPN is initialized by MSRA initialization method [41]. And the RoIAlign [43] is utilized for better localization of bounding boxes. We implement ARDFPN on the publicly Caffe2 platform and train the model on 8 NVIDIA GeForce GTX 1080Ti GPUs with 12GB memory.

B. Detection Comparisons

FPN [11] is one of the most prevalent detectors in many vision tasks, e.g., object detection and instance segmentation. As an augmented version of FPN, ARDFPN is very convenient to replace the original FPN directly in any structures. Hence we compare the proposed ARDFPN with a re-implement
TABLE I Object detection comparisons using Faster R-CNN evaluated on the COCO minival set. The C4 means features extracted from the final convolutional layer of the 4th stage.

<table>
<thead>
<tr>
<th>Faster R-CNN</th>
<th>Backbone</th>
<th>Scale</th>
<th>AP\text{\textsuperscript{box}}</th>
<th>AP\text{\textsuperscript{bbox}}</th>
<th>AP\text{\textsuperscript{small}}</th>
<th>AP\text{\textsuperscript{medium}}</th>
<th>AP\text{\textsuperscript{large}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4 [43]</td>
<td>ResNet-50</td>
<td>600</td>
<td>54.2</td>
<td>33.6</td>
<td>35.7</td>
<td>14.8</td>
<td>8.4</td>
</tr>
<tr>
<td>FPN [11]</td>
<td>ResNet-50</td>
<td>600</td>
<td>57.6</td>
<td>38.6</td>
<td>35.9</td>
<td>18.4</td>
<td>8.9</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>ResNet-50</td>
<td>600</td>
<td>59.2</td>
<td>40.3</td>
<td>37.2</td>
<td>20.5</td>
<td>40.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Faster R-CNN</th>
<th>Backbone</th>
<th>Scale</th>
<th>AP\text{\textsuperscript{box}}</th>
<th>AP\text{\textsuperscript{bbox}}</th>
<th>AP\text{\textsuperscript{small}}</th>
<th>AP\text{\textsuperscript{medium}}</th>
<th>AP\text{\textsuperscript{large}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4 [43]</td>
<td>ResNet-50</td>
<td>800</td>
<td>55.7</td>
<td>37.3</td>
<td>34.8</td>
<td>18.2</td>
<td>38.9</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>ResNet-50</td>
<td>800</td>
<td>60.5</td>
<td>41.4</td>
<td>38.2</td>
<td>22.4</td>
<td>41.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Faster R-CNN</th>
<th>Backbone</th>
<th>Scale</th>
<th>AP\text{\textsuperscript{box}}</th>
<th>AP\text{\textsuperscript{bbox}}</th>
<th>AP\text{\textsuperscript{small}}</th>
<th>AP\text{\textsuperscript{medium}}</th>
<th>AP\text{\textsuperscript{large}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4 [43]</td>
<td>ResNet-101</td>
<td>600</td>
<td>57.3</td>
<td>38.8</td>
<td>36.7</td>
<td>17.1</td>
<td>41.7</td>
</tr>
<tr>
<td>FPN [11]</td>
<td>ResNet-101</td>
<td>600</td>
<td>59.2</td>
<td>41.0</td>
<td>37.7</td>
<td>19.6</td>
<td>41.3</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>ResNet-101</td>
<td>600</td>
<td>60.7</td>
<td>41.5</td>
<td>38.5</td>
<td>20.9</td>
<td>42.0</td>
</tr>
</tbody>
</table>

TABLE II Object detection comparisons using Mask R-CNN evaluated on the COCO minival set.

<table>
<thead>
<tr>
<th>Mask R-CNN</th>
<th>Backbone</th>
<th>Scale</th>
<th>AP\text{\textsuperscript{box}}</th>
<th>AP\text{\textsuperscript{bbox}}</th>
<th>AP\text{\textsuperscript{small}}</th>
<th>AP\text{\textsuperscript{medium}}</th>
<th>AP\text{\textsuperscript{large}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4 [43]</td>
<td>ResNet-50</td>
<td>600</td>
<td>54.8</td>
<td>37.5</td>
<td>34.8</td>
<td>16.3</td>
<td>39.5</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>ResNet-50</td>
<td>600</td>
<td>59.2</td>
<td>41.1</td>
<td>38.0</td>
<td>21.2</td>
<td>41.0</td>
</tr>
</tbody>
</table>

TABLE III Object detection results on the COCO minival set with different depth in the ADB.

<table>
<thead>
<tr>
<th>Depth</th>
<th>AP\text{\textsuperscript{50}}</th>
<th>AP\text{\textsuperscript{75}}</th>
<th>AP\text{\textsuperscript{80}}</th>
<th>AP\text{\textsuperscript{90}}</th>
<th>#params</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPN</td>
<td>57.6</td>
<td>38.6</td>
<td>35.9</td>
<td>18.4</td>
<td>48.7</td>
</tr>
<tr>
<td>1</td>
<td>58.1</td>
<td>39.1</td>
<td>36.3</td>
<td>18.8</td>
<td>49.0</td>
</tr>
<tr>
<td>3</td>
<td>58.6</td>
<td>39.6</td>
<td>36.7</td>
<td>19.4</td>
<td>48.8</td>
</tr>
<tr>
<td>5</td>
<td>58.8</td>
<td>39.8</td>
<td>36.9</td>
<td>20.2</td>
<td>49.6</td>
</tr>
</tbody>
</table>

TABLE IV Object detection results on the COCO minival set with different width in the ADB.

<table>
<thead>
<tr>
<th>Depth</th>
<th>AP\text{\textsuperscript{50}}</th>
<th>AP\text{\textsuperscript{75}}</th>
<th>AP\text{\textsuperscript{80}}</th>
<th>AP\text{\textsuperscript{90}}</th>
<th>#params</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPN</td>
<td>57.6</td>
<td>38.6</td>
<td>35.9</td>
<td>18.4</td>
<td>48.7</td>
</tr>
<tr>
<td>1</td>
<td>57.7</td>
<td>39.0</td>
<td>36.3</td>
<td>19.5</td>
<td>39.3</td>
</tr>
<tr>
<td>3</td>
<td>58.3</td>
<td>39.5</td>
<td>36.5</td>
<td>19.3</td>
<td>39.9</td>
</tr>
<tr>
<td>5</td>
<td>58.6</td>
<td>39.6</td>
<td>36.7</td>
<td>19.4</td>
<td>40.0</td>
</tr>
</tbody>
</table>

TABLE V Object detection results on the COCO minival set with different dilation rates in the aggregated dilation module.

<table>
<thead>
<tr>
<th>Dilation Rate</th>
<th>AP\text{\textsuperscript{50}}</th>
<th>AP\text{\textsuperscript{75}}</th>
<th>AP\text{\textsuperscript{80}}</th>
<th>AP\text{\textsuperscript{90}}</th>
<th>AP\text{\textsuperscript{50}}</th>
<th>AP\text{\textsuperscript{75}}</th>
<th>AP\text{\textsuperscript{80}}</th>
<th>AP\text{\textsuperscript{90}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>2\textsuperscript{w×w}</td>
<td>58.6</td>
<td>39.6</td>
<td>36.7</td>
<td>19.4</td>
<td>40.0</td>
<td>48.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w\textsuperscript{d−1}</td>
<td>58.6</td>
<td>39.5</td>
<td>36.7</td>
<td>19.7</td>
<td>40.0</td>
<td>48.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>w\textsuperscript{d×d}</td>
<td>58.6</td>
<td>39.4</td>
<td>36.7</td>
<td>20.2</td>
<td>40.0</td>
<td>49.1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE VI Ablation study of ARDFPN. Here the result marked in red represents that there is performance improvement over the previous component, while the result marked in blue stands for the performance reduction.

version of FPN in different backbones, scales, and detection frameworks, to show the generalization capability of ARDFPN in this subsection. We also give the basic detector with no feature pyramid as C4 [43], which generates anchors in the final convolution layer of the 4th stage in ResNet [32]. Here, the depth and width of ADBs are set to 3, and the kernel size, padding, and stride of the transposed convolution kernel are set to 8, 3, and 2, respectively. The proposed ARDFPN is built by the output of the last residual blocks from Conv2 to Conv5.

We first compare ARDFPN with the original FPN [11] using the backbone of ResNet-50 with images of scale-600 in the Faster R-CNN detection framework, where the short side of the input images are resized to 600 pixels. The AP\text{\textsuperscript{50}}, AP\text{\textsuperscript{75}}, and AP are already improved for 1.6%, 1.7%, and 1.3%, respectively. More specifically, ARDFPN yields 2.1%, 1.6%, and 0.9% gains in AP\text{\textsuperscript{50}}, AP\text{\textsuperscript{75}}, and AP, which show an excellent performance especially for detecting small and middle objects. Furthermore, by using images of scale-800, the overall performance improvement between the original FPN and ARDFPN are further increased by 2%, 1.8%, and 1.5% in AP\text{\textsuperscript{50}}, AP\text{\textsuperscript{75}}, and AP. We demonstrate that the larger scale of the input image can introduce more contextual cues for the feature generation procedure of ARDFPN. To evaluate the performance of ARDFPN with different backbones, we conduct experiments by using a much deeper network — ResNet-101. As shown in Table I, the total AP\text{\textsuperscript{50}} is still improved for 1.5%, while the AP\text{\textsuperscript{50}} is increased by 1.3%. It should be noticed that the performance of the proposed ARDFPN using ResNet-50 is already comparable with the FPN using ResNet-101, especially for the small objects. Next, we compare ARDFPN with FPN in the Mask R-CNN detection framework in Table. II. By adding the mask branch, the AP\text{\textsuperscript{50}}, AP\text{\textsuperscript{75}}, and AP are gained for 2.1%, 1.6%, and 1.5%, respectively. Compared with the Faster R-CNN framework, the better improvement of ARDFPN when using Mask R-CNN shows the proposed architecture can benefit more with the multi-task learning used in [43]. We give comparisons of detection results between FPN and ARDFPN in Fig. 5.

C. Depth & Width

In this subsection, we evaluate the appropriate depth and width of ADB for ARDFPN. For faster iteration and smaller GPU usage, the experiments are conducted using the backbone of ResNet-50 with images of scale-600 and transposed convolution kernel of (2,0,2) for the following subsections if not otherwise specified. Here FPN [11] is used as the baseline. Table. III shows detection performance of ADB with various depths. To highlight the effect of depth, the dilation rates of ADB are fixed to 1, 2, 4 for each branch in all the level of...
Fig. 6 Evaluation of different sizes of transposed convolution kernels on the COCO \textit{minival}. Since there are large value differences between each metric of COCO, in order to show the changing tendency for all the metrics together, we use a scaled \( mAP \) — \( mAP^* \), which follows \( mAP = \mu + \log_{10} \frac{\mu}{10} \), where \( \mu \) denotes the mean value of each metric for the five kernels. The table below demonstrates the original values.

<table>
<thead>
<tr>
<th>( \times 1 )</th>
<th>( \times 2 )</th>
<th>( \times 4 )</th>
<th>( \times 8 )</th>
<th>( \times 16 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{AP} \text{50}</td>
<td>58.6</td>
<td>59.0</td>
<td>58.9</td>
<td>59.2</td>
</tr>
<tr>
<td>\text{AP} \text{75}</td>
<td>39.4</td>
<td>39.5</td>
<td>40.1</td>
<td>40.3</td>
</tr>
<tr>
<td>\text{AP} \text{50%}</td>
<td>36.7</td>
<td>36.7</td>
<td>36.8</td>
<td>37.2</td>
</tr>
<tr>
<td>\text{AP} \text{1%}</td>
<td>20.2</td>
<td>19.8</td>
<td>20.2</td>
<td>20.5</td>
</tr>
<tr>
<td>\text{AP} \text{all}</td>
<td>40.0</td>
<td>40.3</td>
<td>39.8</td>
<td>40.5</td>
</tr>
<tr>
<td>\text{AP} \text{small}</td>
<td>40.1</td>
<td>40.0</td>
<td>40.0</td>
<td>40.6</td>
</tr>
</tbody>
</table>

Fig. 7 Evaluation of different activation functions adopted in ADBs on the COCO \textit{minival}. Still, we use the \( mAP^* \) to demonstrate the performance variation of different activation functions. The original values are placed on the top of the bars.

ARDFPN. It can be seen that ARDFPN benefits from deeper architecture dramatically. Specifically, with five-layer depth, all of the AP metrics are boosted above 0.6\% compared to the one-layer depth. Considering the trade-off between the accuracy and the model size of ARDFPN, we set the depth to 3 in all the following experiments.

Then we compare the performance of different widths of ADB in Table. IV. Here the dilation rate of each branch in ADB is computed by \( 2^{w-1} \) for simplicity. By increasing the number of the branch from 1 to 3, the \( AP^{50\%} \) is improved for 0.7\% while still maintaining the accuracy of \( AP^* \) and \( AP^\text{l} \). With the help of the improvement for middle size instances, the \( AP^{50\%} \) is gained for about 1.0\%. Since the dilation rates are fixed for all the level of ARDFPN in this experiment, only the instances within a certain scale are benefited from the multi-branch architecture. Thus, we talk about the dilation rate choice in the following subsection to show that ADBs can be conducive to multi-scale instances with depth-based variable dilation rates.

Additionally, we give a discussion about the number of parameters of ARDFPN in terms of the width and depth of ADB. Compared to FPN which uses the legacy interpolation method, ARDFPN employs more dilated convolutional layers to augment the effective receptive fields and exploit the correlations of regions that are adopted for feature pyramid generation. It can be noticed that the performance is improved significantly with gradually increasing depth or width of ADB, where the corresponding number of parameters of ARDFPN is increased linearly. Although the used parameters in ARDFPN are more than the traditional FPN, the overall number is still acceptable for the GPU-based detection scenarios with much better accuracy.

D. Dilation Rate

To further explore the capability of ADB, we exploit the dilation rate choice of each layer in ADB. As shown in Table. V, we give a representative comparison of different chosen scheme of dilation rate, i.e., \( 2^{w-1}, \ 2^{d-1}, \text{ and } w \times d \). First, the dilation rates are changed following \( 2^{w-1} \), which only relates to the width of ADB. As mentioned in the above subsection, by using fixed dilation rates of each branch in ADB, limited contextual information is actually applied into the procedure of pyramidal feature generation for multi-scale instances. In order to capture a large scope of contextual information, we adopt variable dilation rates in each branch, where the dilation rates are increased gradually in cascade. Here, we give two schemes to set the dilation rates in ADB, i.e., \( u^{d-1} \), and \( w \times d \). By enlarging the dilated convolution kernel progressively, the \( AP^* \) are improved substantially while still holding the performance of \( AP^{m} \) and \( AP^{l} \), which demonstrates that the details of small-scale instances can be restored more readily through gradually changing dilation rates. Furthermore, with the moderately increasing scheme — \( w \times d \), the ADB can benefit more from gradual adaptation to the contextual scales, which further enhances the \( AP^* \) and \( AP^{l} \) for 0.8\% and 0.3\% compared to the fixed setting scheme, respectively. It should be noticed that the \( AP^{50\%}, AP^{75\%} \), and \( AP \) are not improved because a slight reduction of \( AP^{m} \) can lead to a relatively large reduction of the total performance since the middle-sized instances take up a large proportion of the dataset in MS COCO [21]. But still, we propose \( w \times d \) as the default dilation rate setting scheme considering the difficulty for small object detection.

E. Transposed Convolution Kernel

In this subsection, we exploit the appropriate transposed convolution kernel of the ADBs. The experiments are executed with the backbone of ResNet-50 with images of scale-600. As shown in Fig. 6, with larger transposed convolution kernel, the detection performance shows an increasing trend, which
demonstrates the effectiveness that making finer feature map correlate with more coarser regions in the enlarging procedure. In particular, the kernel $(8,3,2)$ shows the best detection performance in almost all metrics except the $AP^s$, which lowers by $0.1\%$ with the kernel $(10,4,2)$. Compared with the smallest kernel $(2,0,2)$, which the finer region is only associated with one coarser region, the $AP^{50}$, $AP^{75}$, and $AP$ are all increased more than $0.5\%$. More specifically, the larger transposed convolution kernel benefits more on the instances of middle or large size, which the $AP^m$ and $AP^l$ are improved for $0.5\%$. We argue that, by using a suitable kernel in ADBs, more contextual information can be complemented for pyramidal feature generation. Here, we adopt kernel $(8,3,2)$ as the default setting for ARDFPN.

### F. Activation Function

Since the feature map generation is a non-linear procedure, the interior activation functions play an important role. Typically, different activation functions transform the feature maps in different ways, especially for the negative parts of the feature maps. Here, we compare the detection performance of ARDFPN with four types of activation layers, i.e. ReLU [40], Swish [42], PReLU [41], and Channel-Shared PReLU (CSPReLU) [41]. As illustrated in Fig. 7, the CSPReLU surpasses other activation functions in all of the metrics on the COCO minival with marginal costs, where the most significant improvement comes from the $AP^m$, which increases the precision for about $0.7\%$.

Instead of using channel-wised controlling weights in PReLU, the channel-shared variant transforms the negative parts of the feature maps in the same way with much fewer parameters. We demonstrate that the requirements of activation functions are also different between feature extraction and feature generation. In the feature extraction procedure, the channels of feature maps are tended to be transformed in different ways for much and better semantical representations. However, in the feature generation procedure, the feature maps prefer the homogeneous transformation for each channel with interleaved information. With the transformation sharing mechanism, the negative parts of the feature maps can be effectively activated for contextual information enhancing, which is important for feature generation. We consider that the performance reduction of Swish is caused by its weak activation for the negative parts, where only a small range of negative values are transformed to numerical small non-zero values [42].

### G. Ablation Study

In order to show the performance of each component in ARDFPN, we conduct an ablation study in this subsection to exhibit the contributions by adding components gradually. Here we adopt the FPN [11] as the baseline. The ablation results are illustrated in Table. VI. In the ablation study, we first add three plain convolution layers as shown in Fig. 3(a), each of which is followed by a CSPReLU layer to increase the non-linearity for the original FPN. It can be seen that simply stacking convolutional and nonlinear layers into the feature pyramid even decrease the detection performance. Here, we argue that such a phenomenon is caused by the degradation problem [32]. To address this problem, we propose the transposed residual learning to build the feature pyramid. By utilizing transposed residual learning, all of the metrics start to be improved compared to the baseline. Such improvement reveals that residual learning is vital to the feature pyramid generation.

We then replace the plain convolutions by the ADBs, where the detection precisions are further gained, especially for the small and middle instances. By enlarging the effective receptive fields through multi-branch dilated convolutions, both the $AP^s$ and $AP^m$ are increased for $0.7\%$, which show the effectiveness of the contextual information enhancing. Moreover, with larger transposed convolution kernels, the $AP^m$ and $AP^l$ are further improved by $0.5\%$, which compensates for the deficiency of the above components. As shown in the ablation study, the stronger semantic feature maps can be generated for multi-scale detection by exploiting the correlation of the pyramidal feature regions.

### H. Average Recall

In some practical scenarios, e.g., autonomous driving, the recall rate is more concerned than the precision rate for finding missing objects. Hence we investigate the influence of ARDFPN for average recall rate in this subsection. We conclude the results at different maximum detection number and scales in Table. VII. We first compare the proposed ARDFPN with the original FPN [11] in the Faster R-CNN framework. The ARDFPN exhibits a more powerful capability for finding small and middle objects, which yields $1.6\%$ and $1.3\%$ improvement in $AR^s$ and $AR^m$, respectively.

We then exploit the recall performance of ARDFPN in the Mask R-CNN framework. It can be noticed that the improvement is marginal by simply adding the mask branch [43] into the Faster R-CNN. However, by integrating the ARDFPN with Mask R-CNN, the $AR^s$ and $AR^m$ are further gained for $2.5\%$ and $1.3\%$, respectively. And the total

---

**TABLE VII** Comparison of Average Recall (AR) on the COCO minival set. The $AR^s$, $AR^m$, and $AR^l$ are conducted with 100 maximum detections. Here, the ARDFPN is constructed by using the backbone of ResNet-50 with images of scale-600 and transposed convolution kernel of $(8,3,2)$.

<table>
<thead>
<tr>
<th>Method</th>
<th>$AR^{s=1}$</th>
<th>$AR^{s=10}$</th>
<th>$AR^{s=100}$</th>
<th>$AR^{medium}$</th>
<th>$AR^{large}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>FPN [11]</td>
<td>30.3</td>
<td>47.1</td>
<td>49.3</td>
<td>53.0</td>
<td>63.7</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>30.8</td>
<td>47.9</td>
<td>50.3</td>
<td>54.3</td>
<td>64.5</td>
</tr>
<tr>
<td>FPN [11]</td>
<td>30.6</td>
<td>47.5</td>
<td>49.6</td>
<td>53.3</td>
<td>65.5</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>31.2</td>
<td>48.7</td>
<td>51.2</td>
<td>54.6</td>
<td>66.4</td>
</tr>
</tbody>
</table>

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/.
TABLE VIII Instance segmentation comparisons by using the mask evaluation on the COCO minival set.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Scale</th>
<th>AP\text{IoU=0.50}</th>
<th>AP\text{IoU=0.75}</th>
<th>AP</th>
<th>AP\text{small}</th>
<th>AP\text{medium}</th>
<th>AP\text{large}</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4 [43]</td>
<td>ResNet-50</td>
<td>600</td>
<td>51.0</td>
<td>31.7</td>
<td>30.2</td>
<td>9.8</td>
<td>33.2</td>
</tr>
<tr>
<td>FPN [11]</td>
<td>ResNet-50</td>
<td>600</td>
<td>54.1</td>
<td>34.4</td>
<td>32.9</td>
<td>13.2</td>
<td>35.0</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>ResNet-50</td>
<td>600</td>
<td>56.4</td>
<td>35.7</td>
<td>34.0</td>
<td>14.2</td>
<td>36.4</td>
</tr>
</tbody>
</table>

TABLE IX Comparisons of segmentation metrics on the COCO minival set.

<table>
<thead>
<tr>
<th></th>
<th>Dice</th>
<th>Specificity</th>
<th>Sensitivity</th>
<th>VS</th>
<th>VOE(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C4 [43]</td>
<td>0.714</td>
<td>0.918</td>
<td>0.755</td>
<td>0.823</td>
<td>4.755</td>
</tr>
<tr>
<td>FPN [11]</td>
<td>0.742</td>
<td>0.923</td>
<td>0.778</td>
<td>0.835</td>
<td>4.479</td>
</tr>
<tr>
<td>ARDFPN</td>
<td>0.754</td>
<td>0.932</td>
<td>0.786</td>
<td>0.845</td>
<td>4.423</td>
</tr>
</tbody>
</table>

Fig. 8 Each image doublet shows the comparison of mask results between the proposed ARDFPN (left) and the original FPN (right) on the COCO minival.

AR\text{Det}=100 are increased by 1.6%. We consider that the even better improvement of Mask R-CNN comes from the much more non-linear layers in ARDFPN, which provide more optimization space for the multi-task loss [43]. Since ARDFPN generates pyramidal feature maps by larger receptive fields, the finer layers involve more context cues for detecting small and middle objects.

I. Instance Segmentation

As one of the most crucial and challenging tasks in computer vision, instance segmentation has been received increasing attention in recent years. It aims to predict pixel-wise categories by instance masks to distinguish different objects. In order to testify the flexibility and robustness of the proposed ARDFPN, we conduct an instance segmentation experiment by using the Mask R-CNN framework. Here, we implement the comparisons with the backbone of ResNet-50 and images of scale-600. As shown in Table. VIII, ARDFPN outperforms the original C4 and FPN in all of the metrics without specific hyper-parameters adjustment. In particular, the proposed ARDFPN yields 2.3% improvement in AP\text{IoU=0.50}. We give comparisons of mask results between FPN and ARDFPN in Fig. 8.

Furthermore, we present the comparisons of traditional segmentation metrics in Table. IX to show the performance of ARDFPN for image segmentation. According to [46], [47], [48], we evaluate ARDFPN in five metrics including Dice, Specificity (which are also referred to as True Negative Rate), Sensitivity (which are also referred to as True Positive Rate), Volumetric Similarity (VS), and Volumetric Overlap Error (VOE). It can be seen that ARDFPN exhibits the best performance in all of the metrics. The results of Dice, Specificity, and Sensitivity demonstrate images are segmented with higher IoU for the predicted mask and ground-truth mask by using ARDFPN with Mask R-CNN. Additionally, VS and VOE show that ARDFPN can generate segments with closer volume and more similarity for the ground-truth mask. Experiment results indicate the superiority of ARDFPN in segmentation tasks compared to the original FPN. Since ARDFPN is not designed for image segmentation specifically, we deem that the mentioned performance can be further improved by utilizing methods, like adding the Dice Loss [49] in the multi-task loss function.

V. Conclusion

In this paper, we present a novel detector named Aggregated Residual Dilation based Feature Pyramid Network (ARDFPN) to tackle the multi-scale detection. As an easily extensible module, ARDFPN is simple to implement and train to replace the original FPN in a wide range of flexible architecture designs. In order to enlarge effective receptive fields for pyramidal feature generation, we introduce a multi-branch transformation block based on cascaded dilated convolutions to capture a long-range of contextual information for corresponding scales. A transposed residual learning architecture is proposed to ease the training of the feature pyramid. Moreover, we utilize larger transposed convolution kernels to further enhance the region correlations. Quantitative evaluations showcase the superior performance and generality of the proposed ARDFPN compared to the original FPN through object detection and instance segmentation tasks.

In future work, we will explore the better fusion method for the lateral and top-down features rather than simply using element-wise summation. Besides, how to extract the most useful information from the backbone network for the pyramidal feature generation is also a challenging topic. A promising direction is utilizing the attention method [50] to obtain the region correlation spatially. It should be mentioned...
that the current ARDFPN model still needs a large number of parameters. For this reason, we intend to conduct model compression [51] for ARDFPN in the future, thus achieving comparable performance not just in GPU-based servers, but also in frequently used mobile devices. Another interesting direction for future research is using the multispectral or hyperspectral images or videos to train and inference the proposed ARDFPN for object detection or tracking [52], [53]. Here, we hope the model can be used in more scenarios, including keynote detection [54], action recognition [55], and so on.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant 61525101 and 61227801. We would like to thank Di Wu, Jingran Chen and Kezhong Zhang for their suggestions and comments.

REFERENCES


This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2019.2941892, IEEE Access


**XIAOTONG ZHAO** received the B.S. degree from the Huazhong University of Science and Technology in 2013. He is currently a Ph.D. candidate in communication and information systems with the Wireless Technology Innovation Institute, Beijing University of Posts and Telecommunications. He is also a member of the Key Laboratory of Universal Wireless Communications, Ministry of Education, China. His research interests include object recognition and detection, reinforcement learning and neural architecture search.

**WEI LI** received his Ph.D. degree in Electrical and Computer Engineering from the University of Victoria, Canada in 2004. In 2005 he joined the France Telecom San Francisco Lab as a Research Scientist, where he was a member of international standard bodies for wireless networks. He is currently an assistant professor at the Northern Illinois University. His research interests are in intelligent networks and systems, computer vision, smart grid, Internet of Things, big data analytics and artificial intelligence.

**YIFAN ZHANG** is an associate professor in the School of Information and Communication Engineering of Beijing University of Posts and Telecommunications (BUPT). He received his B.S. degree in BITU, and M.S. and Ph.D. degrees in BUPT. His current research interests include signal modulation classification, signal processing, cognitive radio network implementation, optimization algorithm in wireless networks, and computer vision.

**SHUO CHANG** received the B.S. degree from the Shenyang Jianzhu University of Communication Engineering in 2015. He is currently a Ph.D. candidate in communication and information systems with the Wireless Technology Innovation Institute, Beijing University of Posts and Telecommunications. His research interests include object tracking, image classification and transfer learning.

**ZHIYONG FENG** received the B.S., M.S., and Ph.D. degrees from the Beijing University of Posts and Telecommunications, China. She is currently a Professor with BUPT and the Director of the Key Laboratory of Universal Wireless Communications, Ministry of Education, China. Her main research interests include the wireless network virtualization in fifth generation mobile networks (5G), spectrum sensing and dynamic spectrum management in cognitive wireless networks, universal signal detection and identification, network information theory, and so on. She is active in standards development, such as ITU-R WP5A/WP5D, IEEE 1990, ETSI, and CCSA.

**PING ZHANG** received the Ph.D. degree from the Beijing University of Posts and Telecommunications. He is currently a Professor with the Beijing University of Posts and Telecommunications, China. She is the Vice Director of the Science Bulletin (published by Chinese Academy of Science), an Expert of MIIT major special technological programs, a member of the IMT-Advanced (4G) Group, and a Consultancy Expert of the National Science Foundation Committee. She has authored eight books until now. He was invited to participate in the publishing of Long Term Evolution, and invited to write the book Cognitive Radio Networks (Science Publishing Company, 2010). He has authored over 400 papers at home and abroad publications, such as the IEEE Communications Magazine, the IEEE ELECTRONICS LETTERS, the Chinese Journal of Electronics. He holds about 170 patents. He acts as the Vice Editor-in-Chief of the History of China’s Telecommunication Discipline organized by the China Institute of Communications.