

POD: Practical Object Detection with Scale-Sensitive Network

Junran Peng^{1,2,3}, Ming Sun², Zhaoxiang Zhang^{*1,3}, Tieniu Tan^{1,3}, and Junjie Yan²

¹University of Chinese Academy of Sciences

²SenseTime Group Limited

³Center for Research on Intelligent Perception and Computing, CASIA

Abstract

Scale-sensitive object detection remains a challenging task, where most of the existing methods could not learn it explicitly and are not robust to scale variance. In addition, the most existing methods are less efficient during training or slow during inference, which are not friendly to real-time applications. In this paper, we propose a practical object detection method with scale-sensitive network.

Our method first predicts a global continuous scale, which is shared by all position, for each convolution filter of each network stage. To effectively learn the scale, we average the spatial features and distill the scale from channels. For fast-deployment, we propose a scale decomposition method that transfers the robust fractional scale into combination of fixed integral scales for each convolution filter, which exploits the dilated convolution. We demonstrate it on one-stage and two-stage algorithms under different configurations. For practical applications, training of our method is of efficiency and simplicity which gets rid of complex data sampling or optimize strategy. During testing, the proposed method requires no extra operation and is very supportive of hardware acceleration like TensorRT and TVM. On the COCO test-dev, our model could achieve a 41.5 mAP on one-stage detector and 42.1 mAP on two-stage detectors based on ResNet-101, outperforming baselines by 2.4 and 2.1 respectively without extra FLOPS.

1. Introduction

With the blooming development of CNNs, great progresses have been achieved in the field of object detection. A lot of CNN based detectors are proposed, and they are widely used in real world applications like face detection, pedestrian and vehicle detection in self-driving and augmented reality.

*Corresponding author.

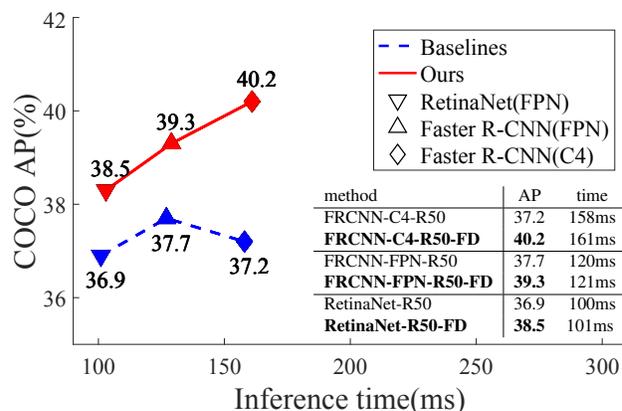


Figure 1: Speed (ms) versus accuracy (AP) on COCO test-dev. Preset with the learnt scales, our fast-deployment(FD) detectors outperforms baselines of one-stage and two-stage detectors by a large margin almost without any extra inference time. All of the models above are trained with $2 \times$ lr schedule. Better results and details are given in §4.

In practical scenarios mentioned above, detection frameworks always work on mobile or embedded devices, and are strictly required to be fast and accurate. While many complicated networks and frameworks are designed to achieve higher accuracy, lots of extra parameters are imported, and larger memory cost and more inference are needed. In this paper we propose a method that could greatly improve the accuracy of detection frameworks without any extra parameters, memory or time cost.

Scale variation is one of the most challenging problems in object detection. In [30], anchor boxes of different sizes are predefined to handle scale variation. Besides, some methods [21, 25] apply astrous convolutions to enlarge receptive field size, which makes information in larger scales captured. In addition, some related works explicitly plug the scale module into backbone. [3] designs a deformable convolution to learn the scale of object to some degree,

which is of local and dense learning style.

However, these methods contain two limitations: 1) many predefined rules based on heuristic knowledge 2) algorithms are hostile to hardware and slow during inference. Many object detection methods are being hampered by these constraints, especially the slow inference on mobile or embedded devices. Methods like [21] and [25] are heavily heuristic-guided and could partly handle scale variation. While the offsets are learned efficiently in deformable convolution [3], the inference is time-consuming due to the bilinear operation at each position. In addition, the grid sampling location in convolution of these methods [3, 41] are dynamic and local, which is not practical in real-time applications for the reason that fixed and integral grid sampling locations are essential for hardware optimization. For instance, a normal convolution network with fixed grid sampling locations could be accelerated on various kinds of devices after optimized by TVM, while convolutions like DCN [3] or SAC [41] are not supported in TVM due to their dynamic and local sampling mechanisms. In this paper, we show that the local and dense continuous scales which are hard to optimize are not necessary and through collaboration of well-learned global scales on layers, a network could be granted the scale-awareness. Therefore, we first train a global scale learner(GSL) network to learn the distribution of continuous scale rates in h and w directions for layers. The scales learnt by our GSL modules are extremely robust across samples and could be treated as fixed values which is revealed in 4.4. Secondly, the learnt arbitrary scales are transformed into combination of fixed integral dilations through our scale decomposition method. Once completing the decomposition, we can transfer the weights and finetune a fast-deployment(FD) network using the decomposed network architecture. The blocks in FD network are granted the desirable combination of dilations which are learnt by GSL network, and the whole network could better cope with objects of a large range of scale while keeping friendly to hardware optimization.

Extensive experiments on the COCO show the effectiveness of our method. With dilations learnt and weights transferred from GSL network, our FD network could outperform Faster R-CNN with C4-ResNet-50 backbone by 3.0% AP and RetinaNet with ResNet-50 by 1.6% AP as shown in Figure 1. Applying our method on two-stage detector based on ResNet101 could achieve 42.1% AP on COCO *test-dev* with a total $2\times$ the training epochs, surpassing its $2\times$ baseline by 2.5% AP. While on RetinaNet based on ResNet-101, our method could yield a 2.4% AP improvement and achieve a 41.5% AP.

Overall, there are several advantages about our method:

- **Large Receptive Fields** : We design a module to learn a stable global scale for each layer and prove that these learnt scales could collaborate to significantly help network han-

dle objects of a large range of scales.

- **Fast Deployment** : We propose a dilation decomposition method to transfer fractional scale into combination of integral dilations. The decomposed dilations which are integral and fixed enable our fast-deployment network to be fast and supportive of hardware optimization during inference.

- **Higher Performance** : Our method is widely applicable across different detection frameworks(both one-stage and two-stage) and network architectures, and could bring significant and consistent gains on both AP50 and AP without being time-consuming. We are confident that our method would be useful in practical scenarios.

2. Related Works

2.1. Deep Learning for Object detection

Object detection is one of the most important and fundamental problems in computer vision area. A variety of effective networks [35, 34, 12, 15, 38, 14, 13, 32] designed for image classification are used as backbone to help the task of object detection. There are generally two types of detectors.

The first type, known as two-stage detector, generates a set of region proposals and classifies each of them by a network. R-CNN [6] generates proposals by Selective Search [37] and extracts features to an independent ConvNet. Later, SPP [11] and Fast-RCNN [5] are proposed that features of different proposals could be extracted from single map through a special pooling layer to reduce computations. Faster R-CNN [30] first proposes a unified end-to-end detectors which introduces a region proposal network(RPN) for feature extraction and proposal generation. There are also many works that extend Faster R-CNN in various aspects to achieve better performance. FPN [22] is designed to fuse features at multiple resolutions and provide anchors specific to different scales. Recent works like Cascade R-CNN [1] and IoU-Net [18] are devoted to increasing the quality of proposals to fit the COCO AP metrics.

The second type, known as one-stage detector, acts like a multi-class RPN that predicts object position and class efficiently within one stage. YOLO [28] outputs bounding box coordinates directly from images. SSD and DSSD involve multi-layer prediction module which helps the detection of objects within different scales. RetinaNet [23] introduces a new focal loss to cope with the foreground-background class imbalance problems. RefineDet [42] adds an anchor refinement module and [20] predicts objects as paired corner keypoints to yield better performance.

2.2. Receptive Fields

The receptive field is a crucial issue in various visual tasks, as output of a neuron only responds to information within its receptive field. Dilated convolution [40] is one of

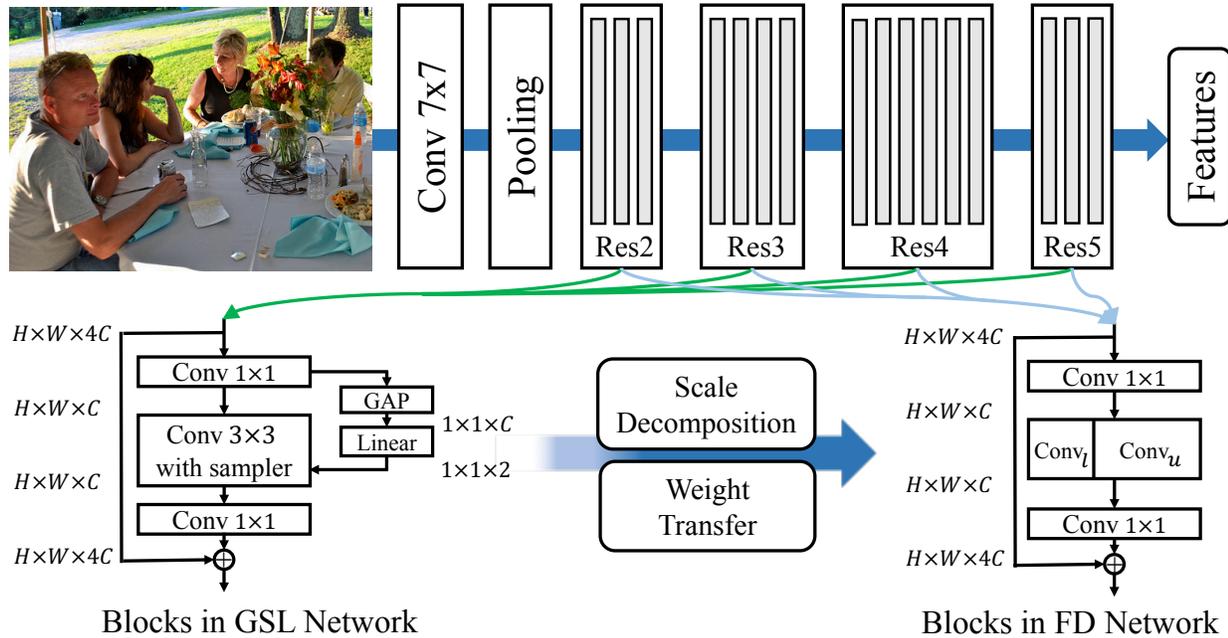


Figure 2: **Overview of the pipeline of our method.** We take ResNet-50 as backbone of detector in this example. We first train a GSL Network to learn a stable scale for each block in h and w directions respectively. The learnt scales are decomposed into combination of integral dilations for each block we care. Given groups of integral dilations, we construct a fast-deployment network and finetune it from GSL network.

the effective solution to enlarge receptive field size, which is widely used in semantic segmentation [2, 43] to incorporate contextual information. In DetNet [21] and RFBNet [25], dilated convolution are used to have larger receptive field size without changing spatial resolution. In [17], convolution learns a fixed deformation of kernel structure from local information.

There are other methods that could automatically learn dynamic receptive fields. In SAC [41], scale-adaptive convolutions are proposed to predict a local flexible-size dilations at each position to cover objects of various sizes. [3] proposes deformable convolution in which the grid sampling locations are offsets predicted with respect to the preceding feature maps. STN [16] introduces a spatial transformer module that can actively transform a feature map by producing an appropriate transformation for each input sample. While effective in performance, these methods are slow and unfriendly to hardware optimization in real-time applications. The framework we present in this paper could keep scale-aware and supportive of hardware optimization at the same time.

3. Our Approach

An overview of our method is illustrated in Figure 2. There are mainly two steps in our approach: 1) We design a global scale learning(GSL) module. Plain blocks are re-

placed with GSL modules during training to learn a recommended global scale. 2) Then we transfer the learnt arbitrary scale into fixed integral dilations using scale decomposition. Equipped with the decomposed integral dilations, a fast-deployment(FD) model is trained with weights transferred from GSL network. In this way we can acquire a high-performance model comprising of convolutions with only fixed and integral dilations, which is friendly to hardware acceleration and fast during inference.

To evaluate capability of handling scale variation of a network, we define **density of connection** as $\mathcal{D}(\Delta p) = \sum \prod_{\Delta p} w$ which means the sum of all paths between a neuron and its input at relative position Δp with all w set to 1. There are usually numerous connections between neurons and each connection is assigned a weight w . A normal network learns the value of weights without changing the architecture of connections during training. We prove in this part that GSL network we design is able to learn scales of each blocks which improves density of connections of the entire network, and that through scale decomposition our FD network could maintain a consistent density of connections with GSL network.

3.1. Global Scale Learner

In our approach, we substitute 3×3 convolutions with our global scale learning modules as shown in Fig 2 in GSL

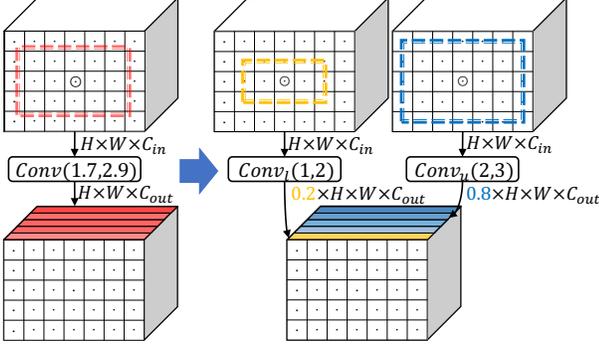


Figure 3: An example of scale decomposition. A 3×3 convolution is split into a sub-conv with lower-bound dilations and a sub-conv with upper-bound dilations through scale decomposition.

network. The scale learning module in our block is split into two parts, a global scale predictor and a convolution with bi-linear sampler which is widely used in methods like DCN[3], SAC[41], STN[16]. The global scale predictor takes the input feature map $U \in \mathbb{R}^{H \times W \times C}$ with width W , height H , and C channels, and feeds it into a global average pooling layer. A linear layer takes the pooled features of channels C and outputs a two-channel global scales in both directions for each image.

Based on the predicted scale (d_h, d_w) , we sample the inputs and pass them to convolution. For example, \mathcal{H} and \mathcal{W} define relative position of inputs with respect to output in a 3×3 convolution with dilations, input and output channels set to 1 for simplicity, which is,

$$\mathcal{H} = \{-1, 0, 1\}, \mathcal{W} = \{-1, 0, 1\}.$$

Given kernel weight w and input feature map x , the output feature at position (h_0, w_0) could be obtained by:

$$y(h_0, w_0) = \sum_{i \in \mathcal{H}, j \in \mathcal{W}} w_{ij} x(h_0 + id_h, w_0 + jd_w).$$

Since coordinates of x at position $(h, w) = (h_0 + id_h, w_0 + jd_w)$ might be fractional, we sample value of input feature $x(h, w)$ through bi-linear interpolation, formulated as:

$$x(h, w) = \sum_{h^*, w^*} \lambda(h^*, h) \lambda(w^*, w) x(h^*, w^*),$$

where $h^* \in \{\lfloor h \rfloor, \lceil h \rceil\}$, $w^* \in \{\lfloor w \rfloor, \lceil w \rceil\}$ and $\lambda(p, q) = \max(0, 1 - |p - q|)$.

Taking ResNet with bottleneck [12] as an example, we apply our GSL module on all the 3×3 conv layers in stage

conv3, *conv4*, and *conv5*. We also try to apply it on *conv2*, but the learnt scales are so close to 1 thus it is meaningless to learn global scale in *conv2*. As mentioned in 4.4, the learnt scales are extremely stable across samples. The standard deviations of predicted global scales on the whole training set are less than 3% of the mean value, while standard deviations of local scales generated in deformable convolution is greater than 50% of the mean value (i.e. 5.3 ± 3.3 for small objects and 8.4 ± 4.5 for large objects in res5c) as mentioned in [3]. Thus we randomly sample 500+ images in training set and are able to treat the average of predicted scales as fixed values.

3.2. Scale Decomposition

In this part, we propose a scale decomposition method that can transform fractional scales into combination into integral dilations for acceleration while keeping **density of connections** consistent. Taking a 1-D convolution (kernel size=3) with a fractional scale rate d as example, it takes input map with C_{in} channels and output C_{out} channels regardless of spatial size. For each location p on the output feature map $\{y_j\}_{j \in C_{out}}$, we have

$$y_j(p) = \sum_{i=1}^{C_{in}} \sum_{\lambda \in \mathcal{L}} w_{ij}^\lambda x_i(p + \lambda d),$$

where λ denotes the relative location in $\mathcal{L} = \{-1, 0, 1\}$.

When d is fractional, convolution with bilinear sampler would both take $x(p + \lambda \lfloor d \rfloor)$ and $x(p + \lambda \lceil d \rceil)$ as inputs with a split factor α for bilinear interpolation defined as $\alpha = d - \lfloor d \rfloor$. Here we have

$$y_j(p) = \sum_{i=1}^{C_{in}} \sum_{\lambda \in \mathcal{L}} w_{ij}^\lambda ((1-\alpha)x_i(p + \lambda \lfloor d \rfloor) + \alpha x_i(p + \lambda \lceil d \rceil)).$$

And the density of connection at relative position $\lambda \lfloor d \rfloor$ and $\lambda \lceil d \rceil$ with respect to output maps $\{y_j\}_{j \in C_{out}}$ are $(1-\alpha)C_{in}C_{out}$ and $\alpha C_{in}C_{out}$ respectively.

In this stage, a convolution with fractional scale is split into two sub-convolutions with dilation $\lambda \lfloor d \rfloor$ and $\lambda \lceil d \rceil$ respectively. The sub-conv with lower bound dilation $\lambda \lfloor d \rfloor$ takes $(1-\alpha)C_{out}$ output channels and the sub-conv with upper bound dilation $\lambda \lceil d \rceil$ takes the other αC_{out} output channels. Then we have output features $\{y_j^*\}_{j \in C_{out}}$ at position p as:

$$y_j^*(p) = \sum_{i=1}^{C_{in}} \sum_{\lambda \in \mathcal{L}} w_{ij}^\lambda x_i(p + \lambda \lfloor d \rfloor),$$

when $j \in [1, (1-\alpha)C_{out}]$, and

$$= \sum_{i=1}^{C_{in}} \sum_{\lambda \in \mathcal{L}} w_{ij}^\lambda x_i(p + \lambda \lceil d \rceil),$$

when $j \in ((1 - \alpha)C_{out}, C_{out}]$.

Through this scale decomposition, the density of connection at relative position $\lambda \lfloor d \rfloor$ and $\lambda \lceil d \rceil$ with respect to output maps $\{y_j^*\}_{j \in C_{out}}$ are $(1 - \alpha)C_{in}C_{out}$ and $\alpha C_{in}C_{out}$ respectively, which are consistent with convolution with fractional scale. In particular, we learn scales in both H and W directions as (d_h, d_w) and set the split factor $\alpha = (d_h - \lfloor d_h \rfloor + d_w - \lfloor d_w \rfloor)/2$ for simplicity¹. An illustration of decomposing a 2-D convolution with learnt scales (1.7, 2.9) is in Figure 3. The split factor is computed as $\alpha = \frac{0.7+0.9}{2} = 0.8$ in this case.

In this way we transfer the interpolation in spatial domain to channel domain while keeping the density of connection between inputs and outputs almost unchanged. We reset dilations of all 3×3 convolutions in conv{3,4,5} using the learnt global scales. For those convs whose learnt scales are fractional, they are replaced with two sub-convs inside as described above. Now that our model consists of only convolutions with fixed integral dilations, it can be easily optimized and be fast in practical scenarios.

There are some special cases that the global scale learnt in h or w direction is less than 1, and the lower-bound dilation is set to 0 after scale decomposition. To handle this issue, we set the kernel size of corresponding direction to 1 which is equivalent to 0 dilation. Besides, if the learnt scale in one direction is very close to integer ($\Delta \leq 0.05$), we do the decomposition on the other direction. For instance, if the learnt dilation is (2.02, 1.7), then the dilations of the two outcome convolutions are set to (2,1) and (2,2) respectively with $\alpha = 0.7$.

3.3. Weight Transfer

Since the density of connections in FD network is consistent with it in GSL network, it is free to take advantage of model weights in GSL network for initialization of FD network. Unlike conventional finetuning which directly copies weights of pretrained models, weights are also decomposed and transferred to layers in FD network in this paper. For convolutions assigned with integer scales, they are initialized with the weights of corresponding layers in pretrained model. For convolutions with fractional scales that are split into two sub-convs with different dilations, the sub-conv with lower-bound dilations takes the weight of the first $(1 - \alpha)$ output channels while the sub-conv with upper dilations takes the weight of the last α output channels from corresponding layer in pretrained model as demonstrated in Fig 4.

In special cases that generates 1×1 , 1×3 and 3×1 sub-convs, weights are initialized with values of corresponding position in 3×3 pretrained convolutions. Fig 5 demonstrates how weights are initialized in different cases.

¹This simplification in 2-D case may change the density of connection a little, but does not matter.

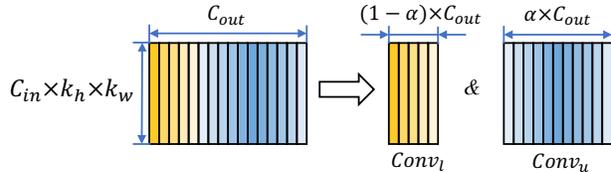


Figure 4: Weight transfer of layers in FD network. When a normal convolution is split into two sub-convs, the convolution with lower-bound dilations takes the first $(1 - \alpha)$ portion of pretrained weights in C_{out} dim for initialization and the convolution with upper-bound dilations takes the last α portion of pretrained weights.

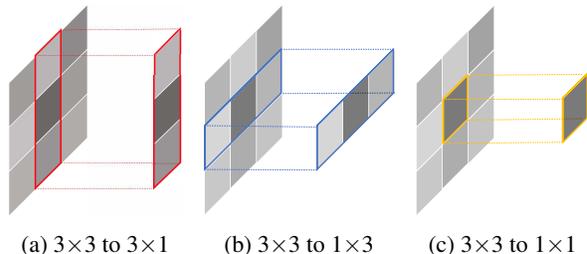


Figure 5: Weight initialization in FD network for different cases of (d_h, d_w) .

4. Experiments

We present experimental results on the bounding box detection track of the challenging MS COCO benchmark [24]. For training, we follow common practice [30] and use the MS COCO trainval35k split (union of 80k images from train and a random 35k subset of images from the 40k image val split). If not specified, we report studies by evaluating on the minival5k split. The COCO-style Average Precision (AP) averages AP across IoU thresholds from 0.5 to 0.95 with an interval of 0.05. These metrics measure the detection performance of various qualities. Final results are also reported on the *test-dev* set.

4.1. Training Details

We adopt the depth 50 or 101 ResNets w/o FPN as backbone of our model. In Faster-RCNN without FPN we adopt C4-backbone and C5-head while in Faster-RCNN with FPN and RetinaNet we adopt C5-backbone. RoI-align is chosen as our pooling strategy in both baselines and our models. We use SGD to optimize the training loss with 0.9 momentum and 0.0001 weight decay. The initial learning rate are set 0.00125 per image for both Faster-RCNN and RetinaNet. By convention, no data augmentations except standard horizontal flipping are used. In our experiments, all models are trained following $1 \times$ schedule, indicating that learning rate is divided by 10 at 8 and 11 epochs with a to-

Method	Backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
YOLOv2 [29]	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD-513 [26]	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8
DSSD-513 [4]	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1
RefineDet512 [42]	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4
RetinaNet [23]	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2
CornerNet [20]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
RetinaNet-FD-WT(ours)	ResNet-101	41.5	62.4	44.9	24.5	44.8	52.9
FRCNN+++ [30]	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9
FRCNN w FPN [22]	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2
FRCNN w TDM [33]	Inception-ResNet-v2	36.8	57.7	39.2	16.2	39.8	52.1
Regionlets [39]	ResNet-101	39.3	59.8	-	21.7	43.7	50.9
Mask R-CNN [9]	ResNet-101	38.2	60.3	41.7	20.1	41.1	50.2
Fitness NMS [36]	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5
Cascade R-CNN [1]	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2
FRCNN-FD-WT(ours)	ResNet-101	42.1	63.4	45.7	21.8	45.1	57.1

Table 1: Comparison with state-of-the-art one-stage and two-stage detectors on COCO *test-dev*.

tal of 13 epochs. We insist that it is responsible to compare different models with the same training schedule, so we list out training epochs in our experiments for fairness. Warming up and Synchronized BatchNorm mechanism [7, 27] are applied in both baselines and our method to make multi-GPU training more stable.

In GSL network, parameters of backbone are initialized by ImageNet [31] pretrained model [19] straightforwardly, other new parameters are initialized by He (MSRA) initialization [10]. In FD network, parameters are initialized in the same way as GSL network if not specified. For FD models whose weights are transferred from GSL models, we add a suffix “WT” to its name for clarity.

4.2. Ablation Study

• Effectiveness of Learnt Scales

The effects of learnt scales are examined from ablations shown in Table 2. Faster-RCNN with C4-B+C5-Head, FPN and RetinaNet are included in this experiment, and ResNet-50 and ResNet-101 are adopted as backbone. In faster-RCNN using C4-backbone and C5-Head, we apply our method on *conv3* and *conv4* of backbones, and on *conv5* in head. Without bells and whistles, Faster-RCNN-FD with C4-backbone and C5-head on ResNet-50/101 could achieve 38.4/40.9 AP, which outperforms baselines by 3.0/2.2 points. As in Faster-RCNN-FD with FPN and RetinaNet-FD, using ResNet-50 as backbone yields 1.7 and 1.5 AP improvement respectively with neither extra parameters nor extra computational costs imported. Even in deeper backbone like ResNet101, the improvement is still considerable. Results in Table 2 show that using combinations of fixed but more reasonable dilations without any other change could effectively improve the performance in tasks of detection. Note that Faster-RCNN-FD with C4-backbone and C5-head benefits a lot, while improvement on detectors with FPN are less but still remarkable.

• GSL Network vs. FD Network

Table 4 shows the performance and speed comparison between GSL network and FD network. Although GSL model can yield a considerable improvement on AP, it is slow and highly unfriendly to hardware acceleration due to unfixed indexing and interpolation operations. FD model keeps almost the same performance and can reach a relatively high speed even without any optimization.

• Weights Transferred from GSL Network

We also investigate the effect of weight transfer from GSL model. The GSL models are trained with $1 \times$ lr schedule to learn global scales. For fairness, we finetune our FD model from GSL model using weight transfer within $1 \times$ lr schedule, and compare with baselines as well as FD models finetuned from ImageNet model with $2 \times$ lr schedule. Table 5 shows the results of Faster-RCNN with R50-C4, R50-FPN structures and of RetinaNet with R50-FPN. Note that FD- $1 \times$ finetuned from GSL model reaches a higher performance than FD- $2 \times$ finetuned from ImageNet model and can outperform baseline- $2 \times$ by a large margin in R50-C4 structure(3.0%AP) and Retina-R50-FPN(1.6%AP). As in Faster-RCNN with FPN, FD- $1 \times$ model finetuned from GSL model can have a close performance to FD- $2 \times$ model finetuned from ImageNet pretrained model. Thus, it is convenient to get a high performance model as $2 \times$ model with the same training epochs in total through this weight transfer strategy.

We also find model finetuned from ImageNet saturates in performance being trained for more than $2 \times$ the epochs which is consistent to findings mentioned in [8]. As shown in Table 5, using weights transferred from GSL network could yield a further improvement on the saturated performance.

Method	Type	Lr schd	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
FRCNN	R50-C4	1×	35.4	55.6	37.9	18.1	39.4	50.3
FRCNN-FD	R50-C4	1×	38.4(3.0)	59.3	41.5	19.8	42.3	55.3
FRCNN	R101-C4	1×	38.7	59.2	41.5	19.6	43.1	54.5
FRCNN-FD	R101-C4	1×	40.9(2.2)	62.1	43.9	21.6	45.1	57.4
FRCNN	R50-FPN	1×	36.2	58.6	38.6	21.0	39.8	47.2
FRCNN-FD	R50-FPN	1×	37.9(1.7)	60.6	40.8	22.3	41.2	49.6
FRCNN	R101-FPN	1×	38.6	60.7	41.7	22.8	42.8	49.6
FRCNN-FD	R101-FPN	1×	40.1(1.5)	62.8	43.3	23.4	44.6	52.3
Retina	R50-FPN	1×	36.0	56.1	38.6	20.4	40.0	48.2
RetinaNet-FD	R50-FPN	1×	37.5(1.5)	57.7	40.2	20.7	41.0	50.3
RetinaNet	R101-FPN	1×	37.5	57.2	40.1	20.5	41.8	50.2
RetinaNet-FD	R101-FPN	1×	38.8(1.3)	59.0	41.7	21.6	42.9	52.6

Table 2: Comparison with baselines on COCO *minival* using dilations decomposed from global scales learnt in GSL network. Our method is applied on different type of detectors with ResNet-50 and ResNet-101, and outperforms baselines in all cases. All models are finetuned from ImageNet pretrained ResNets.

Method	Dataset	Type	Scale Rate	R ₃₁	R ₃₄	R ₄₁	R ₄₆	R ₅₁	R ₅₃
FRCNN	<i>train</i>	R50-C4	h	0.88 ± 0.00	1.30 ± 0.01	1.39 ± 0.02	8.49 ± 0.28	1.34 ± 0.02	3.57 ± 0.02
			w	0.95 ± 0.00	1.28 ± 0.01	1.41 ± 0.02	9.34 ± 0.32	1.20 ± 0.01	3.45 ± 0.03
FRCNN	<i>minival</i>	R50-C4	h	0.88 ± 0.00	1.30 ± 0.01	1.39 ± 0.02	8.44 ± 0.30	1.34 ± 0.02	3.57 ± 0.02
			w	0.95 ± 0.00	1.28 ± 0.01	1.41 ± 0.02	9.31 ± 0.31	1.20 ± 0.01	3.45 ± 0.03
FRCNN	<i>train</i>	R50-FPN	h	0.89 ± 0.00	1.19 ± 0.01	0.77 ± 0.00	1.41 ± 0.01	0.46 ± 0.00	2.81 ± 0.04
			w	0.93 ± 0.00	1.15 ± 0.01	0.82 ± 0.00	1.39 ± 0.01	0.89 ± 0.00	3.68 ± 0.05
RetinaNet	<i>train</i>	R50-FPN	h	0.94 ± 0.00	1.19 ± 0.00	0.80 ± 0.00	1.46 ± 0.02	0.65 ± 0.00	1.75 ± 0.03
			w	0.98 ± 0.00	1.15 ± 0.00	0.85 ± 0.00	1.47 ± 0.02	0.91 ± 0.00	1.53 ± 0.03

Table 3: Distribution of learnt scales for different blocks on different type of detectors or in different data splits.

Settings	Lr schd	Type	AP	Inference time
FRCNN	1x	R50-C4	35.5	158ms
FRCNN-GSL	1x	R50-C4	38.7	269ms
FRCNN-FD	1x	R50-C4	38.4	159ms
FRCNN	1x	R101-C4	38.7	171ms
FRCNN-GSL	1x	R101-C4	40.7	281ms
FRCNN-FD	1x	R101-C4	40.9	172ms
RetinaNet	1x	R50-FPN	36.0	100ms
RetinaNet-GSL	1x	R50-FPN	38.0	150ms
RetinaNet-FD	1x	R50-FPN	37.5	101ms

Table 4: Comparison of performance and inference time between GSL network and FD network without any hardware optimization.

Settings	Lr schd	Type	AP	AP _s	AP _m	AP _l
FRCNN	2x	R50-C4	37.0	18.6	41.3	51.9
FRCNN-FD	2x	R50-C4	39.6	20.7	43.6	56.5
FRCNN-FD-WT	1x	R50-C4	40.0	20.6	44.1	56.8
FRCNN	2x	R50-FPN	37.5	22.1	40.8	48.4
FRCNN-FD	2x	R50-FPN	39.2	23.4	42.5	50.7
FRCNN-FD-WT	1x	R50-FPN	39.1	23.1	42.5	51.0
RetinaNet	2x	R50-FPN	36.7	19.9	40.4	48.4
RetinaNet-FD	2x	R50-FPN	37.8	21.0	41.0	50.7
RetinaNet-FD-WT	1x	R50-FPN	38.3	21.7	41.7	51.1

Table 5: Comparison of models finetuned from 1× GSL models with 1× schedule and models trained with 2× schedule.

4.3. Compared with SOTA

For complete comparison, we evaluate our method on COCO *test-dev* set. We adopt ResNet-101 in Faster-RCNN with C4-B+C5-Head and RetinaNet with FPN as our baselines. When compared with one-stage detectors, we apply exactly the same scale jitter as Retina800 [23] did, and are finetuned from GSL-1x model using 1.5× schedule. The total number of training epochs is 2.5× the normal training epochs which is consistent with Retina800 [23]. While compared with two-stage detectors, we train our model in single scale and use 1× lr schedule without any whistles and bells for fair comparison. Weight of our models are initialized from GSL networks trained with 1× lr schd which are free to use in our method. Results of both one-stage and two stages detectors are shown in Table 1. Applied on various frameworks, our FD-model based on ResNet-101-C4 and Retina-101-FPN could achieve 42.1 and 41.5 AP respectively, which are competitive performances comparing with other state-of-the-art methods. More importantly, instead of introducing complex modules or extra parameters, our method just unlocks the potential of the original detectors themselves while keeping fast. Besides, this method could be easily applied to backbones like ResNext [38] and mobileNetV2 [32] for better performance or speed. It could also be added into many kinds of frameworks like CornerNet [20], and combined with various techniques like Cascade R-CNN [1], thus there is still room for huge improve-

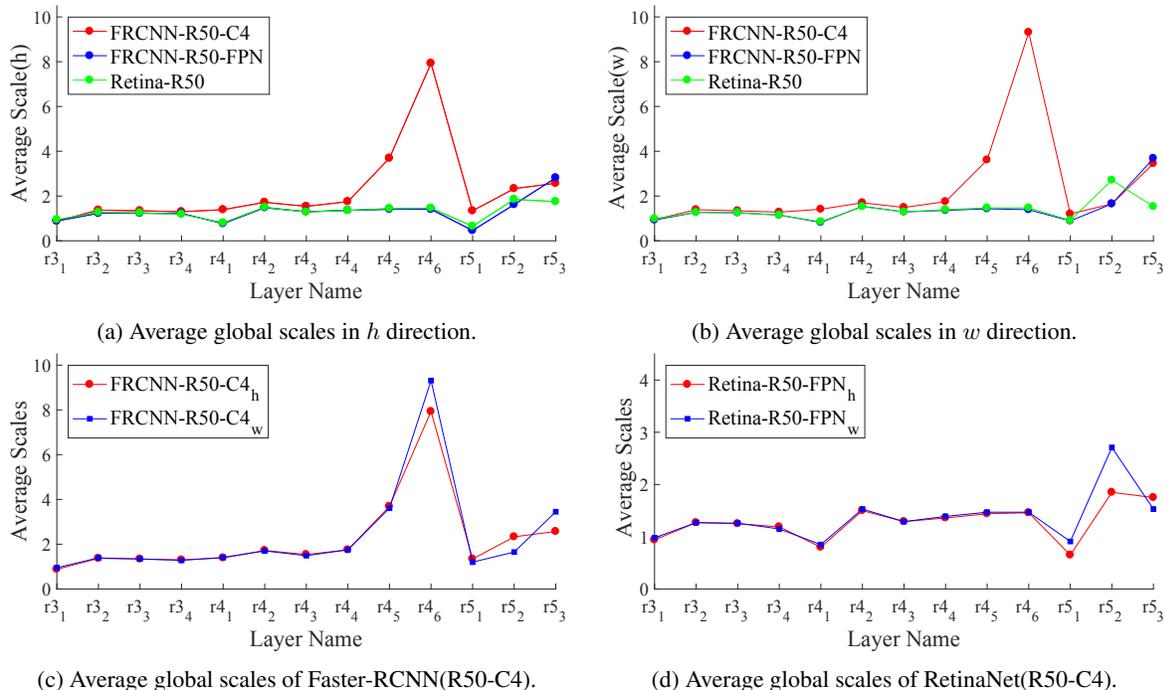


Figure 6: Average of learnt global scales in different detection frameworks.

ment on our method.

4.4. Distribution of Learnt Scales

To better understand our method, we think it is necessary to study to distribution of the learnt scales. Table 3 lists out the mean value of learnt scales of the first and last layers in each stages of different detectors based on ResNet-50. Observations are consistent in other layers. It is interesting to find in the first two rows of table that the learnt scales are very stable across samples and their average of each layer are almost identical in different data splits. This phenomenon demonstrates that it is reliable to use the average of global scales as configured fixed dilations in fast-deployment networks.

Learnt scales of all layers are shown in Figure 6. It is as expected that most layers with stride 1 require larger scale rates. There is also surprising findings that some layers like $res4_6$ in Faster-RCNN with R50-C4 structure demands dilations of (8.49, 9.34) which are huge. Global scales of layers in model with FPN are smaller than those in model without FPN, this is because there are extra down-sampling stages like P5 and P6 (even P7 in Retina) that alleviates the need of a very large receptive field in P4. Another interesting fact is that the learnt scales of convolutions with stride 2 in model with FPN are less than 1 without exception. One explanation is that enlarging receptive field by 2 times during each down-sampling stage is overmuch for FPN which needs to predict objects out from multiple stages, and a less-

than-1 dilation can help shrink the receptive field size.

5. Conclusion

We introduce a practical object detection method to improve scale-sensitivity of detectors. We design a GSL network to learn desirable global scales for different layers in detectors and they collaborate as unity to handle huge scale variation. Then the learnt global scales are decomposed into combination of integral dilations and a FD network is constructed and trained with the decomposed dilations and weights transferred from GSL network. The FD network enables detectors to be scale-sensitive and accurate while keeping fast and supportive of hardware acceleration. Extensive experiments show that our method is effective on various detection frameworks and very useful for practical applications. Since our method is easy to be applied on various frameworks, we will try to combine it with those fastest frameworks to obtain most practical object detectors in future.

6. Acknowledgements

This work was supported in part by the National Key RD Program of China (No.2018YFB-1402605), the Beijing Municipal Natural Science Foundation (No.Z181100008918010), the National Natural Science Foundation of China (No.61836014, No.61761146004, No.61773375, No.61602481). The authors would like thanks NVAAIL for support.

References

- [1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018.
- [2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence(TPAMI)*, 2018.
- [3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, 2017.
- [4] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Amrith Tyagi, and Alexander C Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.
- [5] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, 2015.
- [6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2014.
- [7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large mini-batch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.
- [8] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883*, 2018.
- [9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, 2017.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, 2015.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence(TPAMI)*, 2015.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2016.
- [13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2018.
- [15] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2017.
- [16] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems(NIPS)*, 2015.
- [17] Yunho Jeon and Junmo Kim. Active convolution: Learning the shape of convolution for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2017.
- [18] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems(NIPS)*, 2012.
- [20] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [21] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: Design backbone for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [22] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2017.
- [23] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision(ICCV)*, 2017.
- [24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European conference on computer vision(ECCV)*, 2014.
- [25] Songtao Liu, Di Huang, et al. Receptive field block net for accurate and fast object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *Proceedings of the European conference on computer vision(ECCV)*, 2016.
- [27] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, pages 6181–6189, 2018.
- [28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2016.
- [29] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2017.

- [30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems(NIPS)*, 2015.
- [31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision(IJCV)*, 2015.
- [32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018.
- [33] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond skip connections: Top-down modulation for object detection. *arXiv preprint arXiv:1612.06851*, 2016.
- [34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations(ICLR)*, 2014.
- [35] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2015.
- [36] Lachlan Tychsen-Smith and Lars Petersson. Improving object localization with fitness nms and bounded iou loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018.
- [37] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision(IJCV)*, 2013.
- [38] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2017.
- [39] Hongyu Xu, Xutao Lv, Xiaoyu Wang, Zhou Ren, Navaneeth Bodla, and Rama Chellappa. Deep regionlets for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 798–814, 2018.
- [40] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations(ICLR)*, 2016.
- [41] Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Scale-adaptive convolutions for scene parsing. In *Proceedings of the IEEE International Conference on Computer Vision(ICCV)*, 2017.
- [42] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z Li. Single-shot refinement neural network for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2018.
- [43] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, 2017.