DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution

Siyuan Qiao[1] Liang-Chieh Chen[2] Alan Yuille[1]
[1]Johns Hopkins University
[2]Google Research

Abstract

Many modern object detectors demonstrate outstanding performances by using the mechanism of looking and thinking twice. In this paper, we explore this mechanism in the backbone design for object detection. At the macro level, we propose Recursive Feature Pyramid, which incorporates extra feedback connections from Feature Pyramid Networks into the bottom-up backbone layers. At the micro level, we propose Switchable Atrous Convolution, which convolves the features with different atrous rates and gathers the results using switch functions. Combining them results in DetectoRS, which significantly improves the performances of object detection. On COCO test-dev, DetectoRS achieves state-of-the-art 55.7% box AP for object detection, 48.5% mask AP for instance segmentation, and 50.0% PQ for panoptic segmentation. The code is made publicly available[1].

1. Introduction

To detect objects, human visual perception selectively enhances and suppresses neuron activation by passing high-level semantic information through feedback connections [2, 20, 21]. Inspired by the human vision system, the mechanism of *looking and thinking twice* has been instantiated in computer vision, and demonstrated outstanding performance [5, 6, 62]. Many popular two-stage object detectors, *e.g.*, Faster R-CNN [62], output object proposals first, based on which regional features are then extracted to detect objects. Following the same direction, Cascade R-CNN [5] develops a multi-stage detector, where subsequent detector heads are trained with more selective examples. The success of this design philosophy motivates us to explore it in the neural network backbone design for object detection. In particular, we deploy the mechanism at both the macro and micro levels, resulting in our proposed DetectoRS which significantly improves the performance of the state-of-art object detector HTC [8] by a great margin while a similar inference speed is maintained, as shown in Tab. 1.

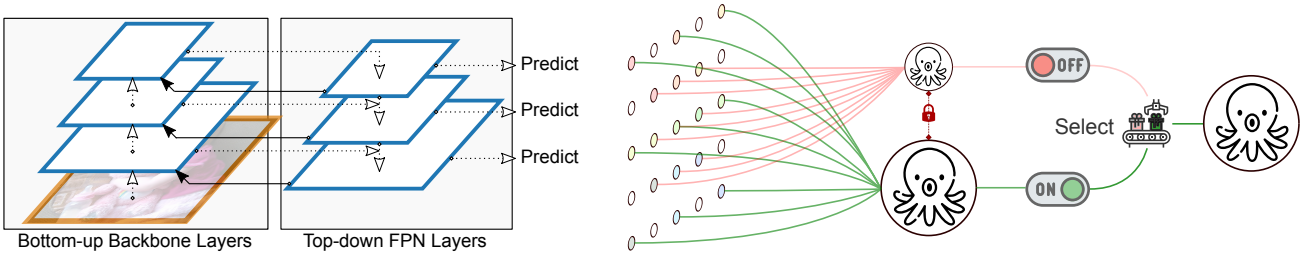
Method	Backbone	AP_{box}	AP_{mask}	FPS
HTC [8]	ResNet-50	43.6	38.5	4.3
DetectoRS	ResNet-50	51.3	44.4	3.9

Table 1: A glimpse of the improvements of the box and mask AP by our DetectoRS on COCO `test-dev`.

At the macro level, our proposed Recursive Feature Pyramid (RFP) builds on top of the Feature Pyramid Networks (FPN) [48] by incorporating extra feedback connections from the FPN layers into the bottom-up backbone layers, as illustrated in Fig. 1a. Unrolling the recursive structure to a sequential implementation, we obtain a backbone for object detector that looks at the images twice or more. Similar to the cascaded detector heads in Cascade R-CNN trained with more selective examples, our RFP recursively enhances FPN to generate increasingly powerful representations. Resembling Deeply-Supervised Nets [39], the feedback connections bring the features that directly receive gradients from the detector heads back to the low levels of the bottom-up backbone to speed up training and boost performance. Our proposed RFP implements a sequential design of *looking and thinking twice*, where the bottom-up backbone and FPN are run multiple times with their output features dependent on those in the previous steps.

At the micro level, we propose Switchable Atrous Convolution (SAC), which convolves the same input feature with different atrous rates [12, 32, 57] and gathers the results using switch functions. Fig. 1b shows an illustration of the concept of SAC. The switch functions are spatially dependent, *i.e.*, each location of the feature map might have different switches to control the outputs of SAC. To use SAC in the detector, we convert all the standard 3x3 convolutional layers in the bottom-up backbone to SAC, which improves the detector performance by a large margin. Some previous methods adopt conditional convolution, *e.g.*, [43, 80], which also combines results of different convolutions as a single output. Unlike those methods whose architecture requires

[1] https://github.com/joe-siyuan-qiao/DetectoRS



(a) Macro Design: Recursive Feature Pyramid.

(b) Micro Design: Switchable Atrous Convolution.

Figure 1: (a) Our Recursive Feature Pyramid adds feedback connections (solid lines) from the top-down FPN layers to the bottom-up backbone layers to look at the image twice or more. (b) Our Switchable Atrous Convolution looks twice at the input features with different atrous rates and the outputs are combined together by switches.

to be trained from scratch, SAC provides a mechanism to easily convert pretrained standard convolutional networks (*e.g.*, ImageNet-pretrained [63] checkpoints). Moreover, a new weight locking mechanism is used in SAC where the weights of different atrous convolutions are the same except for a trainable difference.

Combining the proposed RFP and SAC results in our DetectoRS. To demonstrate its effectiveness, we incorporate DetectoRS into the state-of-art HTC [8] on the challenging COCO dataset [51]. On COCO `test-dev`, we report box AP for object detection [23], mask AP for instance segmentation [28], and PQ for panoptic segmentation [37]. DetectoRS with ResNet-50 [30] as backbone significantly improves HTC [8] by 7.7% box AP and 5.9% mask AP. Additionally, equipping our DetectoRS with ResNeXt-101-64x4d [77] achieves state-of-the-art 55.7% box AP and 48.5% mask AP. Together with the stuff prediction from DeepLabv3+ [15] with Wide-ResNet-41 [11] as backbone, DetectoRS sets a new record of 50.0% PQ for panoptic segmentation.

2. Related Works

Object Detection. There are two main categories of object detection methods: one-stage methods, *e.g.*, [40, 49, 54, 60, 64, 73, 86, 87], and multi-stage methods, *e.g.*, [5, 7, 8, 10, 26, 27, 29, 34, 62, 75]. Multi-stage detectors are usually more flexible and accurate but more complex than one-stage detectors. In this paper, we use a multi-stage detector HTC [8] as our baseline and show comparisons with both categories.

Multi-Scale Features. Our Recursive Feature Pyramid is based on Feature Pyramid Networks (FPN) [48], an effective object detection system that exploits multi-scale features. Previously, many object detectors directly use the multi-scale features extracted from the backbone [4, 54], while FPN incorporates a top-down path to sequentially combine features at different scales. PANet [53] adds another bottom-up path on top of FPN. STDL [88] proposes to exploit cross-scale features by a scale-transfer module. G-FRNet [1] adds feedback with gating units. NAS-FPN [25] and Auto-FPN [79] use neural architecture search [93] to find the optimal FPN structure. EfficientDet [70] proposes to repeat a simple BiFPN layer. Unlike them, our proposed Recursive Feature Pyramid goes through the bottom-up backbone repeatedly to enrich the representation power of FPN. Additionally, we incorporate the Atrous Spatial Pyramid Pooling (ASPP) [14, 15] into FPN to enrich features, similar to the mini-DeepLab design in Seamless [59].

Recursive Convolutional Network. Many recursive methods have been proposed to address different types of computer vision problems, *e.g.*, [35, 46, 69]. Recently, a recursive method CBNet [55] is proposed for object detection, which cascades multiple backbones to output features as the input of FPN. By contrast, our RFP performs recursive computations with proposed ASPP-enriched FPN *included* along with effective fusion modules.

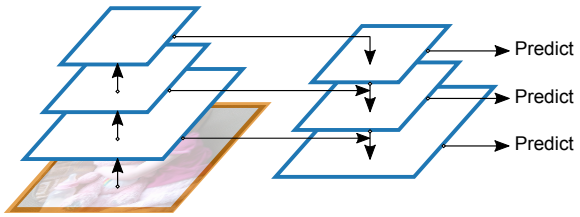
Conditional Convolution Conditional convolutional networks adopt dynamic kernels, widths, or depths, *e.g.*, [17, 43, 47, 52, 80, 83]. Unlike them, our proposed Switchable Atrous Convolution (SAC) allows an effective conversion mechanism from standard convolutions to conditional convolutions without changing any pretrained models. SAC is thus a plug-and-play module for many pretrained backbones. Moreover, SAC uses global context information and a novel weight locking mechanism to make it more effective.

3. Recursive Feature Pyramid

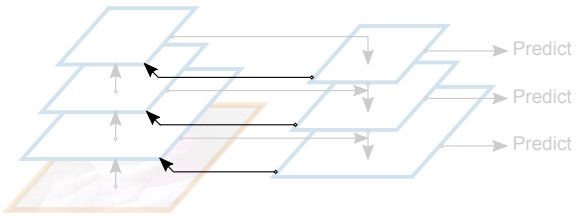
3.1. Feature Pyramid Networks

This subsection provides the background of Feature Pyramid Networks (FPN). Let $\mathbf{B}_i$ denote the i-th stage of the bottom-up backbone, and $\mathbf{F}_i$ denote the i-th top-down FPN operation. The backbone equipped with FPN outputs a set of feature maps $\{\mathbf{f}_i \mid i = 1, ..., S\}$, where S is the number of the stages. For example, $S = 3$ in Fig. 2a. $\forall i = 1, ..., S$, the output feature $\mathbf{f}_i$ is defined by

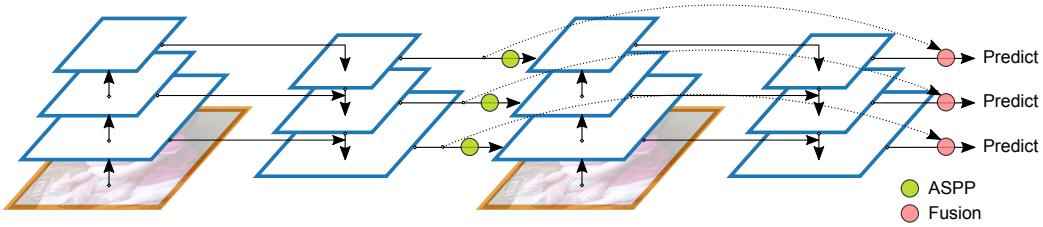
$$\mathbf{f}_i = \mathbf{F}_i\big(\mathbf{f}_{i+1}, \mathbf{x}_i\big), \quad \mathbf{x}_i = \mathbf{B}_i\big(\mathbf{x}_{i-1}\big), \tag{1}$$



(a) FPN.



(b) RFP adds feedback connections to FPN.



(c) Unrolling RFP to an example 2-step sequential implementation.

Figure 2: The architecture of Recursive Feature Pyramid (RFP). (a) Feature Pyramid Networks (FPN). (b) Our RFP incorporates feedback connections into FPN. (c) RFP unrolled to a 2-step sequential network.

where $\mathbf{x}_0$ is the input image and $\mathbf{f}_{S+1} = \mathbf{0}$. The object detector built on FPN uses $\mathbf{f}_i$ for the detection computations.

3.2. Recursive Feature Pyramid

Our proposed Recursive Feature Pyramid (RFP) adds feedback connections to FPN as highlighted in Fig. 2b. Let $\mathbf{R}_i$ denote the feature transformations before connecting them back to the bottom-up backbone. Then, $\forall i = 1, ..., S$, the output feature $\mathbf{f_i}$ of RFP is defined by

$$\mathbf{f}_i = \mathbf{F}_i\big(\mathbf{f}_{i+1}, \mathbf{x}_i\big), \quad \mathbf{x}_i = \mathbf{B}_i\big(\mathbf{x}_{i-1}, \mathbf{R}_i(\mathbf{f}_i)\big), \qquad (2)$$

which makes RFP a recursive operation. We unroll it to a sequential network, *i.e.*, $\forall i = 1, ..., S, t = 1, ...T$,

$$\mathbf{f}_i^t = \mathbf{F}_i^t\big(\mathbf{f}_{i+1}^t, \mathbf{x}_i^t\big), \ \mathbf{x}_i^t = \mathbf{B}_i^t\big(\mathbf{x}_{i-1}^t, \mathbf{R}_i^t(\mathbf{f}_i^{t-1})\big), \qquad (3)$$

where T is the number of unrolled iterations, and we use superscript t to denote operations and features at the unrolled step t. $\mathbf{f}_i^0$ is set to $\mathbf{0}$. In our implementation, $\mathbf{F}_i^t$ and $\mathbf{R}_i^t$ are shared across different steps. We show both shared and different $\mathbf{B}_i^t$ in the ablation study in Sec. 5 as well as the performances with different T's. In our experiments, we use different $\mathbf{B}_i^t$ and set $T = 2$, unless otherwise stated.

We make changes to the ResNet [30] backbone $\mathbf{B}$ to allow it to take both $\mathbf{x}$ and $\mathbf{R}(\mathbf{f})$ as its input. ResNet has four stages, each of which is composed of several similar blocks. We only make changes to the first block of each stage, as shown in Fig. 3. This block computes a 3-layer feature and adds it to a feature computed by a shortcut. To use the feature $\mathbf{R}(\mathbf{f})$, we add another convolutional layer with the kernel size set to 1. The weight of this layer is initialized with $\mathbf{0}$ to make sure it does not have any real effect when we load the weights from a pretrained checkpoint.

3.3. ASPP as the Connecting Module

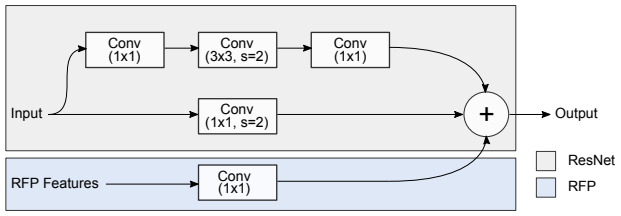


Figure 3: RFP adds transformed features to the first block of each stage of ResNet.

We use Atrous Spatial Pyramid Pooling (ASPP) [13] to implement the connecting module $\mathbf{R}$, which takes a feature $\mathbf{f}_i^t$ as its input and transforms it to the RFP feature used in Fig. 3. In this module, there are four parallel branches that take $\mathbf{f}_i^t$ as their inputs, the outputs of which are then concatenated together along the channel dimension to form the final output of $\mathbf{R}$. Three branches of them use a convolutional layer followed by a ReLU layer, the number of the output channels is $1/4$ the number of the input channels. The last branch uses a global average pooling layer to compress the feature, followed by a 1x1 convolutional layer and a ReLU layer to transform the compressed feature to a $1/4$-size (channel-wise) feature. Finally, it is resized and concatenated with the features from the other three branches. The convolutional layers in those three branches are of the following configurations: kernel size = [1, 3, 3], atrous rate = [1, 3, 6], padding = [0, 3, 6]. Unlike the original ASPP [13],

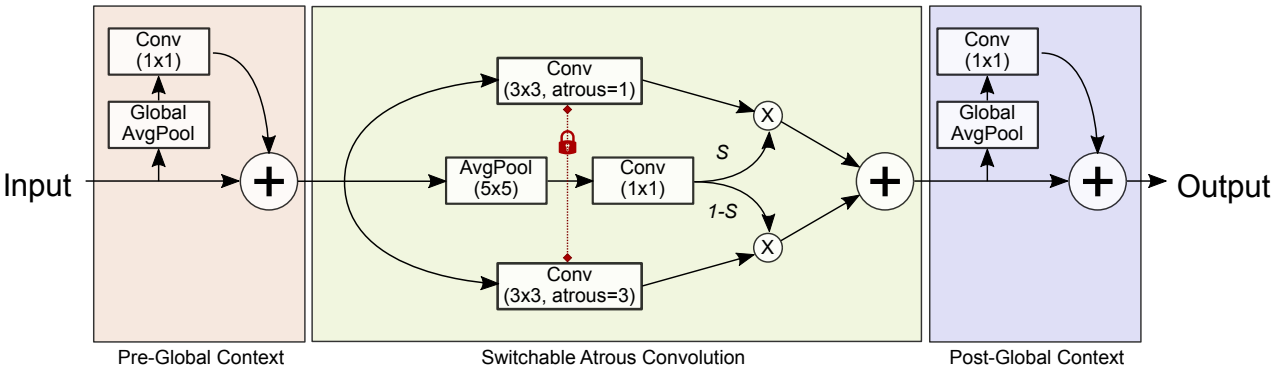


Figure 4: Switchable Atrous Convolution (SAC). We convert every 3x3 convolutional layer in the backbone ResNet to SAC, which softly switches the convolutional computation between different atrous rates. The **lock** indicates that the weights are the same except for a trainable difference (see Eq. 4). Two global context modules add image-level information to the features.

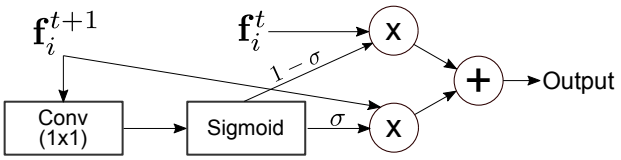


Figure 5: The fusion module used in RFP. σ is the output of Sigmoid, which is used to fuse features from different steps.

we do not have a convolutional layer following the concatenated features as in here $\mathbf{R}$ does not generate the final output used in dense prediction tasks. Note that each of the four branches yields a feature with channels $1/4$ that of the input feature, and concatenating them generates a feature that has the same size as the input feature of $\mathbf{R}$. In Sec. 5, we show the performances of RFP with and without ASPP module.

3.4. Output Update by the Fusion Module

As shown in Fig. 2c, our RFP additionally uses a fusion module to combine $\mathbf{f}_i^t$ and $\mathbf{f}_i^{t+1}$ to update the values of $\mathbf{f}_i$ at the unrolled stage $t+1$ used in Equ. (3). The fusion module is very similar to the update process in recurrent neural networks [31] if we consider $\mathbf{f}_i^t$ as a sequence of data. The fusion module is used for unrolled steps from 2 to T. At the unrolled step $t+1$ ($t=1,...,T-1$), the fusion module takes the feature $\mathbf{f}_i^t$ at the step t and the feature $\mathbf{f}_i^{t+1}$ newly computed by FPN at the step $t+1$ as its input. The fusion module uses the feature $\mathbf{f}_i^{t+1}$ to compute an attention map by a convolutional layer followed by a Sigmoid operation. The resulting attention map is used to compute the weighted sum of $\mathbf{f}_i^t$ and $\mathbf{f}_i^{t+1}$ to form an updated $\mathbf{f}_i$. This $\mathbf{f}_i$ will be used as $\mathbf{f}_i^{t+1}$ for the computation in the following steps. In the ablation study in Sec. 5, we will show the performances of RFP with and without the fusion module.

4. Switchable Atrous Convolution

4.1. Atrous Convolution

Atrous convolution [12, 32, 57] is an effective technique to enlarge the field-of-view of filters at any convolutional layer. In particular, atrous convolution with atrous rate r introduces $r-1$ zeros between consecutive filter values, equivalently enlarging the kernel size of a $k \times k$ filter to $k_e = k + (k-1)(r-1)$ without increasing the number of parameters or the amount of computation. Fig. 1b shows an example of a 3x3 convolutional layer with the atrous rate set to 1 (red) and 2 (green): the same kind of object of different scales could be roughly detected by the same set of convolutional weights using different atrous rates.

4.2. Switchable Atrous Convolution

In this subsection, we present the details of our proposed Switchable Atrous Convolution (SAC). Fig. 4 shows the overall architecture of SAC, which has three major components: two global context modules appended *before* and *after* the SAC component. This subsection focuses on the main SAC component in the middle and we will explain the global context modules afterwards.

We use $\mathbf{y} = \mathbf{Conv}(\mathbf{x}, \mathbf{w}, r)$ to denote the convolutional operation with weight $\mathbf{w}$ and atrous rate r which takes $\mathbf{x}$ as its input and outputs $\mathbf{y}$. Then, we can convert a convolutional layer to SAC as follows.

$$\begin{aligned}\mathbf{Conv}(\mathbf{x}, \mathbf{w}, 1) \xrightarrow[\text{to SAC}]{\text{Convert}} \mathbf{S}(\mathbf{x}) \cdot \mathbf{Conv}(\mathbf{x}, \mathbf{w}, 1) \\ + (1 - \mathbf{S}(\mathbf{x})) \cdot \mathbf{Conv}(\mathbf{x}, \mathbf{w} + \Delta\mathbf{w}, r)\end{aligned} \tag{4}$$

where r here is a hyper-parameter of SAC, $\Delta\mathbf{w}$ is a trainable weight, and the switch function $\mathbf{S}(\cdot)$ is implemented as an average pooling layer with a 5x5 kernel followed by a 1x1 convolutional layer (see Fig. 4). The switch function is input

HTC	RFP	SAC	Box AP	Box AP_{50}	Box AP_{75}	Box AP_S	Box AP_M	Box AP_L	Mask AP	Mask AP_{50}	Mask AP_{75}	Mask AP_S	Mask AP_M	Mask AP_L	Runtime FPS
✓			42.0	60.8	45.5	23.7	45.5	56.4	37.1	58.2	39.9	19.1	40.2	51.9	4.3
✓	✓		46.2	65.1	50.2	27.9	50.3	60.3	40.4	62.5	43.5	22.3	43.8	54.9	4.1
✓		✓	46.3	65.8	50.2	27.8	50.6	62.4	40.4	63.1	43.4	22.7	44.2	56.4	4.2
✓	✓	✓	49.0	67.7	53.0	30.1	52.6	64.9	42.1	64.8	45.5	23.9	45.6	57.8	3.9

Table 2: Detection results on COCO `val2017` with ResNet-50 as backbone. The models are trained for 12 epochs.

and location dependent; thus, the backbone model is able to adapt to different scales as needed. We set $r = 3$ in our experiments, unless stated otherwise.

We propose a locking mechanism by setting one weight as $\mathbf{w}$ and the other as $\mathbf{w} + \Delta\mathbf{w}$ for the following reasons. Object detectors usually use pretrained checkpoints to initialize the weights. However, for an SAC layer converted from a standard convolutional layer, the weight for the larger atrous rate is missing. Since objects at different scales can be roughly detected by the same weight with different atrous rates, it is natural to initialize the missing weights with those in the pretrained model. Our implementation uses $\mathbf{w} + \Delta\mathbf{w}$ for the missing weight where $\mathbf{w}$ is from the pretrained checkpoint and $\Delta\mathbf{w}$ is initialized with $\mathbf{0}$. When fixing $\Delta\mathbf{w} = 0$, we observe a drop of 0.1% AP. But $\Delta\mathbf{w}$ alone without the locking mechanism degrades AP a lot.

4.3. Global Context

As shown in Fig. 4, we insert two global context modules before and after the main component of SAC. These two modules are light-weighted as the input features are first compressed by a global average pooling layer. The global context modules are similar to SENet [33] except for two major differences: (1) we only have one convolutional layer without any non-linearity layers, and (2) the output is added back to the main stream instead of multiplying the input by a re-calibrating value computed by Sigmoid. Experimentally, we found that adding the global context information before the SAC component (*i.e.*, adding global information to the switch function) has a positive effect on the detection performance. We speculate that this is because $\mathbf{S}$ can make more stable switching predictions when global information is available. We then move the global information outside the switch function and place it before and after the major body so that both **Conv** and $\mathbf{S}$ can benefit from it. We did not adopt the original SENet formulation as we found no improvement on the final model AP. In the ablation study in Sec. 5, we show the performances of SAC with and without the global context modules.

4.4. Implementation Details

In our implementation, we use deformable convolution [19, 92] to replace both of the convolutional operations in Eq. 4. The offset functions of them are not shared, which are initialized to predict $\mathbf{0}$ when loading from a pretrained backbone. Experiments in Sec. 5 will show performance comparisons of SAC with and without deformable convolution. We adopt SAC on ResNet and its variants [30, 77] by replacing all the 3x3 convolutional layers in the backbone. The weights and the biases in the global context modules are initialized with $\mathbf{0}$. The weight in the switch $\mathbf{S}$ is initialized with $\mathbf{0}$ and the bias is set to $\mathbf{1}$. $\Delta\mathbf{w}$ is initialized with $\mathbf{0}$. The above initialization strategy guarantees that when loading the backbone pretrained on ImageNet [63], converting all the 3x3 convolutional layers to SAC will not change the output before taking any steps of training on COCO [51].

5. Experiments

5.1. Experimental Details

We conduct experiments on COCO dataset [51]. All the models presented in the paper are trained on the split of `train2017` which has 115k labeled images. Then, we test the models on `val2017` and `test-dev`. We implement DetectoRS with mmdetection [9]. Our baseline model is HTC [8], which uses the bounding box and instance segmentation annotations from the dataset. Runtime is measured on a single NVIDIA TITAN RTX graphics card. We strictly follow the experimental settings of HTC [8]. For ablation studies, we train models for 12 epochs with the learning rate multiplied by 0.1 after 8 and 12 epochs. Additionally, other training and testing settings are kept the same and no bells and whistles are used for them. For our main results after the ablation studies, we use multi-scale training with the long edge set to 1333 and the short edge randomly sampled from [400, 1200]. We train the models for 40 epochs with the learning rate multiplied by 0.1 after 36 and 39 epochs. Soft-NMS [3] is used for ResNeXt-101-32x4d and ResNeXt-101-64x4d. We also report the results with and without test-time augmentation (TTA), which includes horizontal flip and multi-scale testing with the short edge set to [800, 1000, 1200, 1400, 1600] and the long edge set to 1.5x short edge.

5.2. Ablation Studies

In this subsection, we show the ablation studies of RFP and SAC in Tab. 2 and Tab. 3. Tab. 2 shows the box and

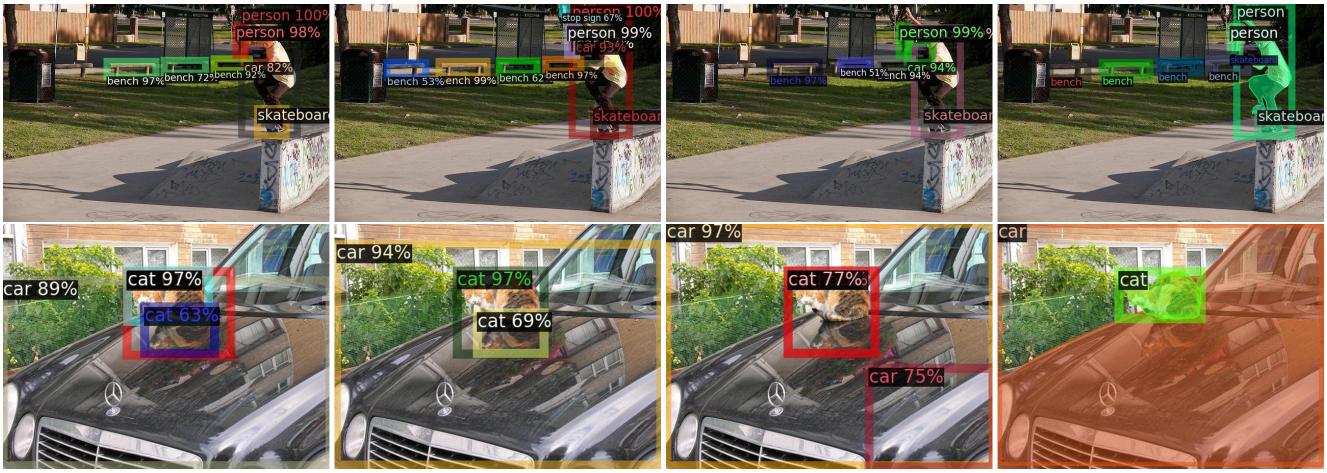


Figure 6: From left to right: visualization of the detection results by HTC, 'HTC + RFP', 'HTC + SAC' and the ground truth.

	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline HTC	42.0	60.8	45.5	23.7	45.5	56.4
RFP	46.2	65.1	50.2	27.9	50.3	60.3
RFP + sharing	45.4	64.1	49.4	26.5	49.0	60.0
RFP - aspp	45.7	64.2	49.6	26.7	49.3	60.5
RFP - fusion	45.9	64.7	50.0	27.0	50.1	60.1
RFP + 3X	47.5	66.3	51.8	29.0	51.6	61.9
SAC	46.3	65.8	50.2	27.8	50.6	62.4
SAC - DCN	45.3	65.0	49.3	27.5	48.7	60.6
SAC - DCN - global	44.3	63.7	48.2	25.7	48.0	59.6
SAC - DCN - locking	44.7	64.4	48.7	26.0	48.7	59.0
SAC - DCN + DS	45.1	64.6	49.0	26.3	49.3	60.1

Table 3: Ablation study of RFP (the middle group) and SAC (the bottom group) on COCO `val2017` with ResNet-50.

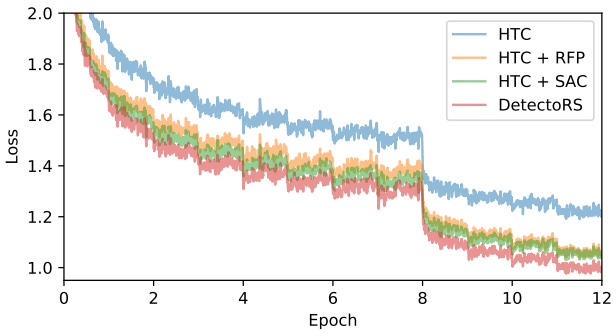


Figure 7: Comparing training losses of HTC, 'HTC + RFP', 'HTC + SAC', and DetectoRS during 12 training epochs.

mask AP of the baseline HTC with ResNet-50 and FPN as its backbone. Then, we add our proposed RFP and SAC to the baseline HTC, both of which are able to improve AP by $> 4\%$ without too much decrease in the speed. Combining them together results in our DetectoRS which achieves 49% box AP and 42.1% mask AP at 3.9 fps.

Tab. 3 shows the individual ablation study of RFP and SAC where we present the sources of their improvements. For RFP, we show 'RFP + sharing' where B_i^1 and B_i^2 share their weights. We also demonstrate the improvements of the ASPP module and the fusion module by presenting the performance of RFP without them as in 'RFP - aspp' and 'RFP - fusion'. Finally, we increase the unrolled step T from 2 to 3 and get 'RFP + 3X', which further improves the box AP by 1.3%. For SAC, we first experiment with SAC without DCN [19] (*i.e.*, 'SAC - DCN'). Then, we show that the global context is able to bring improvements on AP in 'SAC - DCN - global'. 'SAC - DCN - locking' breaks the locking mechanism in Fig. 4 where the second convolution uses only $\Delta\mathbf{w}$, proving that weight locking is necessary for SAC. Finally, in 'SAC - DCN + DS (dual-switch)', we replace $\mathbf{S}(x)$ and $1 - \mathbf{S}(x)$ with two independent switches $\mathbf{S}_1(x)$ and $\mathbf{S}_2(x)$. The ablation study in Tab. 3 shows that the formulations of RFP and SAC have the best configuration within the design space we have explored.

Fig. 6 provides visualization of the results by HTC, 'HTC + RFP' and 'HTC + SAC'. From this comparison, we notice that RFP, similar to human visual perception that selectively enhances or suppresses neuron activations, is able to find occluded objects more easily for which the nearby context information is more critical. SAC, because of its ability to increase the field-of-view as needed, is more capable of detecting large objects in the images. This is also consistent with the results of SAC shown in Tab. 2 where it has a higher AP_L. Fig. 7 shows the training losses of HTC, 'HTC + RFP', 'HTC + SAC', and DetectoRS. Both are able to significantly accelerate the training process and converge to lower losses.

Method	Backbone	TTA	AP_{bbox}	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
YOLOv3 [61]	DarkNet-53		33.0	57.9	34.4	18.3	25.4	41.9
RetinaNet [50]	ResNeXt-101		40.8	61.1	44.1	24.1	44.2	51.2
RefineDet [85]	ResNet-101	✓	41.8	62.9	45.7	25.6	45.1	54.1
CornerNet [38]	Hourglass-104	✓	42.1	57.8	45.3	20.8	44.8	56.7
ExtremeNet [90]	Hourglass-104	✓	43.7	60.5	47.0	24.1	46.9	57.6
FSAF [91]	ResNeXt-101	✓	44.6	65.2	48.6	29.7	47.1	54.6
FCOS [71]	ResNeXt-101		44.7	64.1	48.4	27.6	47.5	55.6
CenterNet [89]	Hourglass-104	✓	45.1	63.9	49.3	26.6	47.1	57.7
NAS-FPN [25]	AmoebaNet		48.3	-	-	-	-	-
SEPC [74]	ResNeXt-101		50.1	69.8	54.3	31.3	53.3	63.7
SpineNet [22]	SpineNet-190		52.1	71.8	56.5	35.4	55.0	63.6
EfficientDet-D7 [70]	EfficientNet-B6		52.2	71.4	56.3	-	-	-
EfficientDet-D7x (Model Zoo on GitHub)	-	-	55.1	74.3	59.9	37.2	57.9	68.0
Mask R-CNN [29]	ResNet-101		39.8	62.3	43.4	22.1	43.2	51.2
Cascade R-CNN [5]	ResNet-101		42.8	62.1	46.3	23.7	45.5	55.2
Libra R-CNN [56]	ResNeXt-101		43.0	64.0	47.0	25.3	45.6	54.6
DCN-v2 [92]	ResNet-101	✓	46.0	67.9	50.8	27.8	49.1	59.5
PANet [53]	ResNeXt-101		47.4	67.2	51.8	30.1	51.7	60.0
SINPER [66]	ResNet-101	✓	47.6	68.5	53.4	30.9	50.6	60.7
SNIP [65]	Model Ensemble	✓	48.3	69.7	53.7	31.4	51.6	60.7
TridentNet [45]	ResNet-101	✓	48.4	69.7	53.5	31.8	51.3	60.3
Cascade Mask R-CNN [5]	ResNeXt-152	✓	50.2	68.2	54.9	31.9	52.9	63.5
TSD [68]	SENet154	✓	51.2	71.9	56.0	33.8	54.8	64.2
MegDet [58]	Model Ensemble	✓	52.5	-	-	-	-	-
CBNet [55]	ResNeXt-152	✓	53.3	71.9	58.5	35.5	55.8	66.7
HTC [8]	ResNet-50		43.6	62.6	47.4	24.8	46.0	55.9
HTC	ResNeXt-101-32x4d		46.4	65.8	50.5	26.8	49.4	59.6
HTC	ResNeXt-101-64x4d		47.2	66.5	51.4	27.7	50.1	60.3
HTC + DCN [19] + multi-scale training	ResNeXt-101-64x4d		50.8	70.3	55.2	31.1	54.1	64.8
DetectoRS	ResNet-50		51.3	70.1	55.8	31.7	54.6	64.8
DetectoRS	ResNet-50	✓	53.0	72.2	57.8	35.9	55.6	64.6
DetectoRS	ResNeXt-101-32x4d		53.3	71.6	58.5	33.9	56.5	66.9
DetectoRS	ResNeXt-101-32x4d	✓	54.7	73.5	60.1	37.4	57.3	66.4
DetectoRS	ResNeXt-101-64x4d	✓	55.7	74.2	61.1	37.7	58.4	68.1

Table 4: State-of-the-art comparison on COCO `test-dev` for bounding box object detection. TTA: test-time augmentation, which includes multi-scale testing, horizontal flipping, *etc*. The input size of DetectoRS without TTA is (1333, 800).

5.3. Main Results

In this subsection, we show the main results of DetectoRS. We equip the state-of-art detector HTC with DetectoRS, and use ResNet-50 and ResNeXt-101 as the backbones for DetectoRS. The bounding box detection results are shown in Tab. 4. The results are divided into 4 groups. The first group shows one-stage detectors. The second group shows multi-stage detectors. The third group is HTC, which is the baseline of DetectoRS. The fourth group is our results. The results can be also categorized as simple test results and TTA results, where TTA is short for test-time augmentation. The third column shows whether TTA is used. Note that different methods use different TTA strategies. For example, CBNet uses a strong TTA strategy, which can improve their box AP from 50.7% to 53.3%. Our TTA strategy only brings 1.4% improvement when using ResNeXt-101-32x4d as backbone. The simple test settings can also vary significantly among different detectors. DetectoRS uses (1333, 800) as the test image size. Larger input sizes tend to bring improvements (see [70]). DetectoRS adopts the same setting of HTC.

We also show the instance segmentation results in Tab. 5. As many methods in Tab. 4 do not provide mask AP in their paper, we only compare DetectoRS with its baseline HTC. The experimental settings for bounding box and mask object detection are the same except that we report AP_{mask} instead of AP_{bbox}. From Tab. 5, we can see that consistent with the bounding box results, DetectoRS also brings significant improvements over its baseline for instance segmentation.

Finally, the panoptic segmentation results are presented in Tab. 6. As DetectoRS only detects things, we use the stuff predictions by DeepLabv3+ [15] with backbone Wide-ResNet-41 [11, 76, 84]. Combing the thing and the stuff predictions using the script available in panoptic API [37] without tuning any hyper-parameters, we set a new state-of-

Figure 8: Visualizing the outputs of the learned switch functions in Switchable Atrous Convolution. Darker intensity means that the switch function for that region gathers more outputs from the larger atrous rate.

Method	Backbone	TTA	AP_{mask}	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
HTC [8]	ResNet-50		38.5	60.1	41.7	20.4	40.6	51.2
HTC	ResNeXt-101-32x4d		40.7	63.2	44.1	22.0	43.3	54.2
HTC	ResNeXt-101-64x4d		41.3	63.9	44.8	22.7	44.0	54.7
HTC + DCN [19] + multi-scale training	ResNeXt-101-64x4d		44.2	67.8	48.1	25.3	47.2	58.7
DetectoRS	ResNet-50		44.4	67.7	48.3	25.6	47.5	58.3
DetectoRS	ResNet-50	✓	45.8	69.8	50.1	29.2	48.3	58.2
DetectoRS	ResNeXt-101-32x4d		45.8	69.2	50.1	27.4	48.7	59.6
DetectoRS	ResNeXt-101-32x4d	✓	47.1	71.1	51.6	30.3	49.5	59.6
DetectoRS	ResNeXt-101-64x4d	✓	48.5	72.0	53.3	31.6	50.9	61.5

Table 5: Instance segmentation comparison on COCO `test-dev`.

Method	TTA	PQ	PQ^{Th}	PQ^{St}
DeeperLab [81]		34.3	37.5	29.6
SSAP [24]	✓	36.9	40.1	32.0
Panoptic-DeepLab [18]	✓	41.4	45.1	35.9
Axial-DeepLab-L [72]	✓	44.2	49.2	36.8
TASCNet [41]		40.7	47.0	31.0
Panoptic-FPN [36]		40.9	48.3	29.7
AdaptIS [67]	✓	42.8	53.2	36.7
AUNet [44]		46.5	55.8	32.5
UPSNet [78]	✓	46.6	53.2	36.7
Li *et al.* [42]		47.2	53.5	37.7
SpatialFlow [16]	✓	47.3	53.5	37.9
SOGNet [82]	✓	47.8	-	-
DetectoRS	✓	50.0	58.5	37.2

Table 6: State-of-the-art comparison on COCO `test-dev` for panoptic segmentation.

the-art of 50.0% PQ for panoptic segmentation on COCO.

5.4. Visualizing Learned Switches

Fig. 8 shows the visualization results of the outputs of the last switch function of 'SAC - DCN' in Tab. 3. Darker intensity in the figure means that the switch function for that region gathers more outputs from the larger atrous rate. Comparing the switch outputs with the original images, we observe that the switch outputs are well aligned with the ground-truth object scales. These results prove that the behaviours of Switchable Atrous Convolution are consistent with our intuition, which tend to use larger atrous rates when encountering large objects.

6. Conclusion

In this paper, motivated by the design philosophy of looking and thinking twice, we have proposed DetectoRS, which includes Recursive Feature Pyramid and Switchable Atrous Convolution. Recursive Feature Pyramid implements thinking twice at the macro level, where the outputs of FPN are brought back to each stage of the bottom-up backbone through feedback connections. Switchable Atrous Convolution instantiates looking twice at the micro level, where the inputs are convolved with two different atrous rates. DetectoRS is tested on COCO for object detection, instance segmentation and panoptic segmentation. It sets new state-of-the-art results on all these tasks.

Acknowledgements

The animal and lock icons are made by Freepik from Flaticon. The switch icon is made by Pixel perfect from Flaticon. The express icon is made by Nhor Phai from Flaticon.

References

[1] Md Amirul Islam, Mrigank Rochan, Neil DB Bruce, and Yang Wang. Gated feedback refinement network for dense image labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3751–3759, 2017. 2

[2] Diane M Beck and Sabine Kastner. Top-down and bottom-up mechanisms in biasing competition in the human brain. *Vision Research*, 49(10):1154–1165, 2009. 1

[3] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-nms–improving object detection with one line of code. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5561–5569, 2017. 5

[4] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *Proceedings of the European Conference on Computer Vision*, pages 354–370. Springer, 2016. 2

[5] 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*, pages 6154–6162, 2018. 1, 2, 7

[6] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2956–2964, 2015. 1

[7] Jiale Cao, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. D2det: Towards high quality object detection and instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11485–11494, 2020. 2

[8] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019. 1, 2, 5, 7, 8

[9] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 5

[10] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2018. 2

[11] Liang-Chieh Chen, Raphael Gontijo Lopes, Bowen Cheng, Maxwell D. Collins, Ekin D. Cubuk, Barret Zoph, Hartwig Adam, and Jonathon Shlens. Naive-student: Leveraging semi-supervised learning in video sequences for urban scene segmentation. In *Proceedings of the European Conference on Computer Vision*, 2020. 2, 7

[12] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *Proceedings of the International Conference on Learning Representations*, 2015. 1, 4

[13] 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*, 40(4):834–848, 2017. 3

[14] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2

[15] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision*, 2018. 2, 7

[16] Qiang Chen, Anda Cheng, Xiangyu He, Peisong Wang, and Jian Cheng. Spatialflow: Bridging all tasks for panoptic segmentation. *arXiv preprint arXiv:1910.08787*, 2019. 8

[17] Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2

[18] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 8

[19] 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*, pages 764–773, 2017. 5, 6, 7, 8

[20] Robert Desimone. Visual attention mediated by biased competition in extrastriate visual cortex. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 353(1373):1245–1255, 1998. 1

[21] Robert Desimone and John Duncan. Neural mechanisms of selective visual attention. *Annual Review of Neuroscience*, 18(1):193–222, 1995. 1

[22] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. *arXiv preprint arXiv:1912.05027*, 2019. 7

[23] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2015. 2

[24] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 642–651, 2019. 8

[25] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019. 2, 7

[26] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015. 2

[27] Chaoxu Guo, Bin Fan, Qian Zhang, Shiming Xiang, and Chunhong Pan. Augfpn: Improving multi-scale feature learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12595–12604, 2020. 2

[28] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 297–312. Springer, 2014. 2

[29] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017. 2, 7

[30] 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*, pages 770–778, 2016. 2, 3, 5

[31] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. 4

[32] Matthias Holschneider, Richard Kronland-Martinet, Jean Morlet, and Ph Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets: Time-Frequency Methods and Phase Space*, pages 289–297. Springer Berlin Heidelberg, 1989. 1, 4

[33] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7132–7141, 2018. 5

[34] Chenhan Jiang, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Sp-nas: Serial-to-parallel backbone search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11863–11872, 2020. 2

[35] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1637–1645, 2016. 2

[36] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 8

[37] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9404–9413, 2019. 2, 7

[38] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision*, pages 734–750, 2018. 7

[39] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial Intelligence and Statistics*, pages 562–570, 2015. 1

[40] Hengduo Li, Zuxuan Wu, Chen Zhu, Caiming Xiong, Richard Socher, and Larry S Davis. Learning from noisy anchors for one-stage object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10588–10597, 2020. 2

[41] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff. *arXiv preprint arXiv:1812.01192*, 2018. 8

[42] Qizhu Li, Xiaojuan Qi, and Philip HS Torr. Unifying training and inference for panoptic segmentation. *arXiv preprint arXiv:2001.04982*, 2020. 8

[43] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 510–519, 2019. 1, 2

[44] Yanwei Li, Xinze Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7026–7035, 2019. 8

[45] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-aware trident networks for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6054–6063, 2019. 7

[46] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375, 2015. 2

[47] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, pages 2181–2191, 2017. 2

[48] 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*, pages 2117–2125, 2017. 1, 2

[49] 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*, pages 2980–2988, 2017. 2

[50] 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*, pages 2980–2988, 2017. 7

[51] 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, pages 740–755. Springer, 2014. 2, 5

[52] Lanlan Liu and Jia Deng. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2

[53] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018. 2, 7

[54] 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*, pages 21–37. Springer, 2016. 2

[55] Yudong Liu, Yongtao Wang, Siwei Wang, TingTing Liang, Qijie Zhao, Zhi Tang, and Haibin Ling. Cbnet: A novel composite backbone network architecture for object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. 2, 7

[56] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2019. 7

[57] George Papandreou, Iasonas Kokkinos, and Pierre-Andre Savalle. Modeling local and global deformations in deep learning: Epitomic convolution, multiple instance learning, and sliding window detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1, 4

[58] 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*, pages 6181–6189, 2018. 7

[59] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8277–8286, 2019. 2

[60] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017. 2

[61] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 7

[62] 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*, pages 91–99, 2015. 1, 2

[63] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 2, 5

[64] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013. 2

[65] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3578–3587, 2018. 7

[66] Bharat Singh, Mahyar Najibi, and Larry S Davis. Sniper: Efficient multi-scale training. In *Advances in Neural Information Processing Systems*, pages 9310–9320, 2018. 7

[67] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7355–7363, 2019. 8

[68] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 7

[69] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3147–3155, 2017. 2

[70] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. *arXiv preprint arXiv:1911.09070*, 2019. 2, 7

[71] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9627–9636, 2019. 7

[72] Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. *arXiv preprint arXiv:2003.07853*, 2020. 8

[73] Ning Wang, Yang Gao, Hao Chen, Peng Wang, Zhi Tian, Chunhua Shen, and Yanning Zhang. Nas-fcos: Fast neural architecture search for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11943–11951, 2020. 2

[74] Xinjiang Wang, Shilong Zhang, Zhuoran Yu, Litong Feng, and Wayne Zhang. Scale-equalizing pyramid convolution for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 7

[75] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10186–10195, 2020. 2

[76] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 2019. 7

[77] 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*, pages 1492–1500, 2017. 2, 5

[78] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pages 8818–8826, 2019. 8

[79] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6649–6658, 2019. 2

[80] Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. In *Advances in Neural Information Processing Systems*, pages 1307–1318. Curran Associates, Inc., 2019. 1, 2

[81] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. Deeperlab: Single-shot image parser. *arXiv preprint arXiv:1902.05093*, 2019. 8

[82] Yibo Yang, Hongyang Li, Xia Li, Qijie Zhao, Jianlong Wu, and Zhouchen Lin. Sognet: Scene overlap graph network for panoptic segmentation. *arXiv preprint arXiv:1911.07527*, 2019. 8

[83] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 2

[84] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference*, 2016. 7

[85] 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*, pages 4203–4212, 2018. 7

[86] Zhishuai Zhang, Siyuan Qiao, Cihang Xie, Wei Shen, Bo Wang, and Alan L Yuille. Single-shot object detection with enriched semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5813–5821, 2018. 2

[87] Qijie Zhao, Tao Sheng, Yongtao Wang, Zhi Tang, Ying Chen, Ling Cai, and Haibin Ling. M2det: A single-shot object detector based on multi-level feature pyramid network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9259–9266, 2019. 2

[88] Peng Zhou, Bingbing Ni, Cong Geng, Jianguo Hu, and Yi Xu. Scale-transferrable object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 528–537, 2018. 2

[89] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 7

[90] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 850–859, 2019. 7

[91] Chenchen Zhu, Yihui He, and Marios Savvides. Feature selective anchor-free module for single-shot object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 840–849, 2019. 7

[92] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 5, 7

[93] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2