arXiv:1711.07264v2 [cs.CV] 23 Nov 2017

Light-Head R-CNN: In Defense of Two-Stage Object Detector

Zeming Li[1], Chao Peng[2], Gang Yu[2], Xiangyu Zhang[2], Yangdong Deng[1], Jian Sun[2]
[1]School of Software, Tsinghua University, {lizm15@mails.tsinghua.edu.cn, dengyd@tsinghua.edu.cn }
[2] Megvii Inc. (Face++), {pengchao, yugang, zhangxiangyu, sunjian}@megvii.com

Abstract

In this paper, we first investigate why typical two-stage methods are not as fast as single-stage, fast detectors like YOLO [26, 27] and SSD [22]. We find that Faster R-CNN [28] and R-FCN [17] perform an intensive computation after or before RoI warping. Faster R-CNN involves two fully connected layers for RoI recognition, while R-FCN produces a large score maps. Thus, the speed of these networks is slow due to the heavy-head design in the architecture. Even if we significantly reduce the base model, the computation cost cannot be largely decreased accordingly.

We propose a new two-stage detector, Light-Head R-CNN, to address the shortcoming in current two-stage approaches. In our design, we make the head of network as light as possible, by using a thin feature map and a cheap R-CNN subnet (pooling and single fully-connected layer). Our ResNet-101 based light-head R-CNN outperforms state-of-art object detectors on COCO while keeping time efficiency. More importantly, simply replacing the backbone with a tiny network (e.g, Xception), our Light-Head R-CNN gets 30.7 mmAP at 102 FPS on COCO, significantly outperforming the single-stage, fast detectors like YOLO [26, 27] and SSD [22] on both speed and accuracy. Code will be made publicly available.

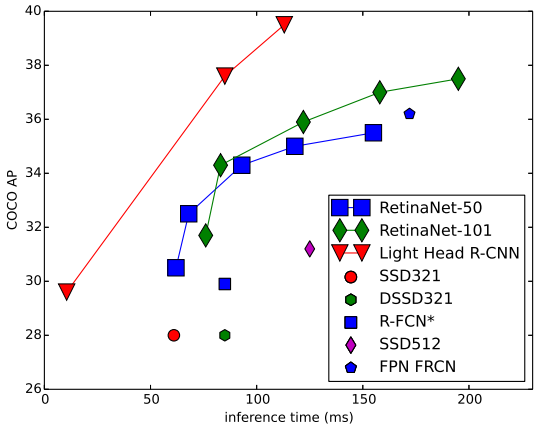

Figure 1. Comparisons of Light Head R-CNN along with previous one-stage and two-stage detectors. We show our results with different backbones (a small Xception like network, Resnet-50, Resnet-101). Thanks for better design principle, our Light Head R-CNN significant outperform all competitors, and provide a new upper envelope. Note that all results reported here are obtained by use single-scale training only. The multi-scale training results are presented in Table 5.

1. Introduction

Recent CNN-based object detectors can be categorized into single-stage detectors [26, 27, 22, 20, 4] and two-stage [5, 28, 19, 7] detectors. The single-stage detector usually targets on a sweet-spot of very fast speed and reasonably good accuracy. The two-stage detector divides the task into two steps: the first step (*body*) generates many proposals, and the second step (*head*) focuses on the recognition of the proposals. Usually, in order to achieve the best accuracy, the design of the head is heavy. The two-stage detector often has a sweet-spot of (relatively) slow speed and very high accuracy.

Could the two-stage detector beat the single-stage detector on both efficiency and accuracy? We find that typical two-stage object detectors like Faster R-CNN [5] and R-FCN [17] share the similar characteristics: a heavy head attached to the backbone network. For example, Faster R-CNN employs two large fully connected layers or all the convolution layers in ResNet 5-th stage [28, 29] for per RoIs recognition and regression. It is time-consuming in terms of per-region prediction and even gets worse when a large number of proposals are utilized. In addition, the number of feature channels after ROI pooling is large, which makes the first fully connected consume a large memory and potentially influence the computational speed. Different from Fast/Faster R-CNN which apply a per-region subnetwork many times, Region-based fully convolutional networks (R-FCN) [17] tries to share computation across all RoIs. However, R-FCN needs to produce a very large additional score maps with $\#classes \times p \times p$ (p is the followed

pooling size) channels, which is also memory and time consuming. The heavy head design of Faster R-CNN or R-FCN make the two-stage approach less competitive if we use a small backbone network.

In this paper, we propose a light-head design to build an efficient yet accurate two-stage detector. Specifically, we apply a large-kernel separable convolution to produce "thin" feature maps with small channel number ($\alpha \times p \times p$ is used in our experiments and $\alpha \leq 10$). This design greatly reduces the computation of following RoI-wise subnetwork and makes the detection system memory-friendly. A cheap single fully-connected layer is attached to the pooling layer, which well exploits the feature representation for classification and regression.

Because of our light-head structure, our detector is able to strike the best tradeoff of speed and accuracy, not matter a large or small backbone network is used. As shown in Figure 1, our algorithm, dotted as Light-Head R-CNN, can significantly outperform the fast single-stage detector like SSD [22] and YOLOv2 [27] with even faster computational speed. In addition, our algorithm is also flexible to large backbone network. Based on a ResNet-101 backbone, we can outperform state-of-art algorithms including two-stage detectors like Mask R-CNN [7] and one-stage detectors like RetinaNet [20].

2. Related works

Benefited from the rapid development of deep convolutional networks[16, 32, 34, 9, 15, 38, 12, 39, 12, 14, 11], a great progress has been made for the object detection problem. We briefly review some of the recent object detection work in two dimensions as follows:

Accuracy perspective: R-CNN [6] is among the first to utilize deep neural network features into detection system. Hand-engineered methods, such as Selective Search [37], Edge Boxes [40], MCG[1], is involved to generate proposals for R-CNN. Then Fast R-CNN [5] is proposed to join train object classification and bounding box regression, which improves the performance by multi-task training. Following Fast R-CNN, Faster R-CNN [28] introduces Region Proposal Network (RPN) to generate proposals by using network features. Benefited from richer proposals, it marginally increase the accuracy. Faster R-CNN has been seen as a milestone of R-CNN serials detectors. Most of the following works strengthen Faster R-CNN by bringing more computation into network. Dai *et al.* propose Deformable Convolutional Networks [3] to model geometric transformations by learning additional offsets without supervision. Lin *et al.* propose Feature Pyramid Networks (FPN) [19], which exploits inherent multi-scale, pyramidal hierarchy of deep convolutional networks to construct feature pyramids. Based on FPN, Mask R-CNN [7] further extends a mask predictor by adding a extra branch in parallel with the bounding box recognition. RetinaNet [20] is another FPN based single stage detector, which involves Focal-Loss to address class imbalance issue caused by extreme foreground-background ratio.

Speed perspective: Object detection literatures have been also strive to improve the speed of the detectors. Back to original R-CNN, which forwards each proposal separately through the whole network. He *et al.* proposes SPP-net [8] to share the computations among candidate boxes. Both Fast/Faster R-CNN [5, 28] accelerates the network by uniforming the detection pipeline. R-FCN [17] shares computations between RoI subnetworks, which speed up inference when a large number of proposals are utilized. Another hot research topic is proposal free detector. YOLO and YOLO v2 [26, 27] simplifies object detection as a regression problem, which directly predicts the bounding boxes and associated class probabilities without proposal generation. SSD [22] further improves performance by producing predictions of different scales from different layers. Unlike box-center based detectors, DeNet [36] first predicts all boxes corners, and then quickly searching the corner distribution for non-trivial bounding boxes.

In conclusion, from accuracy perspective, both one and two-stage detectors have achieved state-of-art precision with nearly speed. However from speed perspective, object detection literature is lack of competitive fast two-stage detector compared single-stage approaches with nearly accuracy. In this paper, we try to design a better and faster two-stage detector called Light head R-CNN to fill this missing.

3. Our Approach

In this section, we will first present our light-head R-CNN and then describe other design details in object detection.

3.1. Light-Head R-CNN

As we discuss in Section 1, conventional two-stage object detectors usually involve a heavy head, which has negative influence on the computational speed. "Head" in our paper refers to the structure attached to our backbone base network. More specifically, there will be two components: R-CNN subnet and ROI warping.

3.1.1 R-CNN subnet

Faster R-CNN adopts a powerful R-CNN which utilizes two large fully connected layers or whole Resnet stage 5 [28, 29] as a second stage classifier, which is beneficial to the detection performance. Therefore Faster R-CNN and its extensions perform leading accuracy in the most challenging benchmarks like COCO. However, the computation could be intensive especially when the number of object proposals is large. To speed up RoI-wise subnet, R-FCN first produces

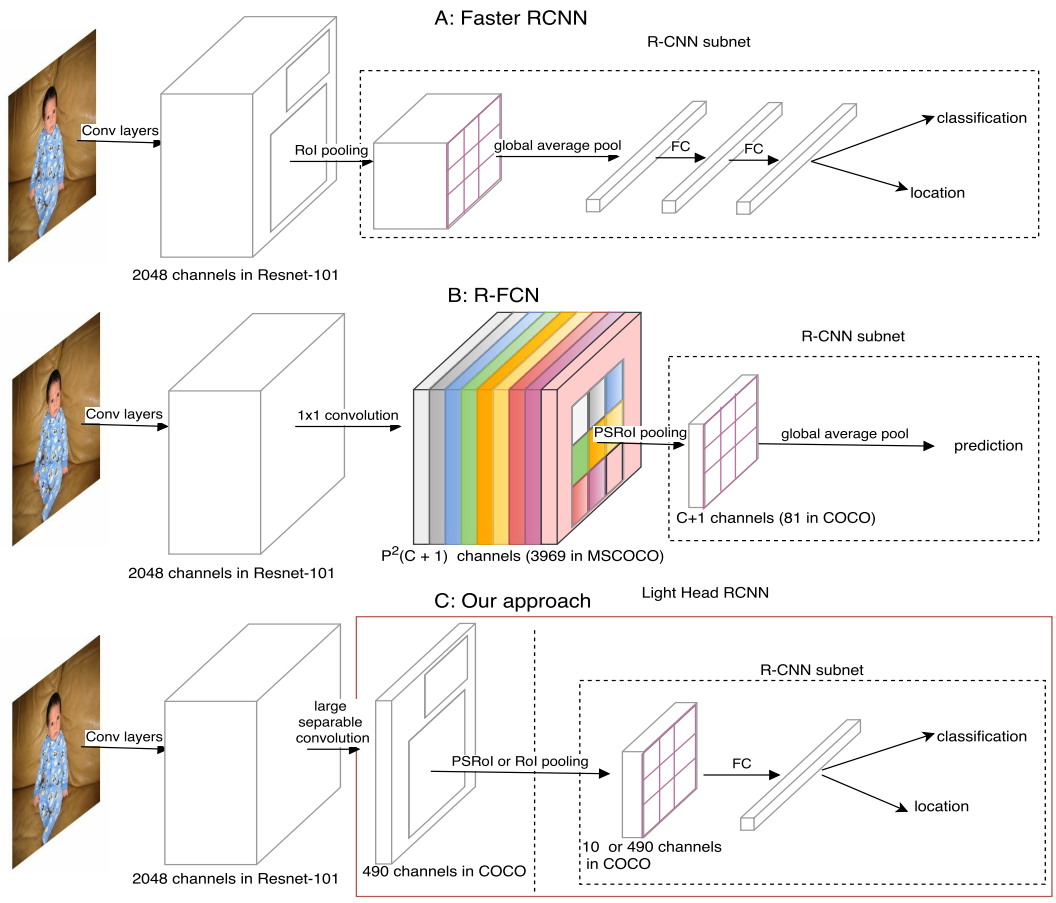

Figure 2. Overview of our approach. Our Light-Head R-CNN builds "thin" feature maps before RoI warping, by large separable convolution. We adopt a single fully-connected layer with 2048 channels in our R-CNN subnet. Thanks for thinner feature maps and cheap R-CNN subnet, the whole network is highly efficient while keeping accuracy.

a set of score maps for each region, whose channel number will be $\#classes \times p \times p$ (p is the followed pooling size), and then pool along each RoI and average vote the final prediction. Using a computation-free R-CNN subnet, R-FCN gets comparable results by involving more computation on RoI shared score maps generation.

As mentioned above, Faster R-CNN and R-FCN have heavy head but at different positions. From **accuracy** perspective, although Faster R-CNN is good at RoI classification, it usually involves global average pooling in order to reduce the computation of first fully connected layer, which is harmful for spatial localization. For R-FCN, it directly pools the prediction results after the position-sensitive pooling and the performance is usually not as strong as Faster R-CNN without RoI-wise computation layers. From **speed** perspective, Faster R-CNN pass every RoI independently through a costly R-CNN subnet, which slows down the network speed especially when the number of proposals is large. R-FCN uses cost-free R-CNN subnet as a second stage detector. But as R-FCN needs to produce a very large score map for RoI pooling, the whole network still time/memory consuming.

Having these issues in mind, in our new Light-Head R-CNN, we propose to utilize a simple, cheap fully-connected layer for our R-CNN subnet, which makes a good trade-off between the performance and computational speed. Figure 2 (C) provides the overview of our Light Head R-CNN. As the computation and memory cost for the fully-connected layer also depends on the number channel maps after ROI operation, we next discuss how we design the ROI warping.

3.1.2 Thin feature maps for RoI warping

Before feeding proposals into R-CNN subnet, RoI warping is involved to make the shape of feature maps fixed.

In our Light-Head R-CNN, we propose to generate the feature maps with small channel number (thin feature

maps), followed by conventional RoI warping. In our experiments, we find that RoI warping on **thin feature maps** will not only improves the accuracy but also saves memory and computation during training and inference. Considering PSRoI pooling on thin feature maps, we can bring more computation to strengthen R-CNN and decrease the channels. In addition, if we apply RoI pooling on our thin feature maps, we can reduce R-CNN overhead and abandon Global Average Pooling to improve performance simultaneously. Moreover, without losing time efficiency, large convolution can be involved for thin feature map production.

3.2. Light-Head R-CNN for Object Detection

Following the above discussions, we present our implementation details for general object detection. The pipeline of our approach is in Figure 2 (C). We have two settings: 1) setting "L" to validate the performance our algorithm when integrated with a large backbone network; 2) setting "S" to validate the effectiveness and efficiency of our algorithm when uses a small backbone network. Unless otherwise specified, Setting L and Setting S share the same other settings.

Basic feature extractor. For the setting L, we adopt ResNet 101 [9] as our basic feature extractor. On the other hand, we utilize the Xception-like small base model for the setting S. The network structure of the Xception model can be found in Table 7. In Figure 2, "Conv layers" indicate our base-model. The last convolution blocks of conv4 and conv5 are denoted as C_4, C_5.

Thin feature maps. We apply large separable convolution layers [35, 25] on C_5, the structure is shown in Figure 3 C. In our approach, we let k to 15, $C_{mid} = 64$ for setting S, and $C_{mid} = 256$ for setting L. We also reduce the C_{out} to $10 \times p \times p$ which is extremely small compared with limited $\#classes \times p \times p$ used in R-FCN. Benefit from the larger valid receptive field caused by large kernel, the feature maps we pooled on are more powerful.

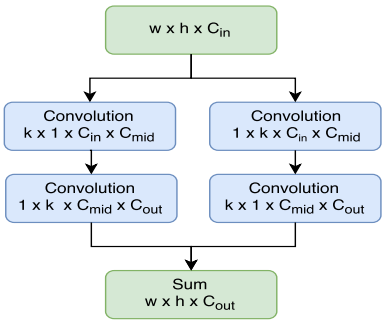

Figure 3. Large separable convolution performs a $k \times 1$ and $1 \times k$ convolution sequentially. The computational complexity can be further controlled through C_{mid}, C_{out}.

R-CNN subnet. Here we only employ a single fully-connected layer with 2048 channels (no dropout) in R-CNN subnet, followed by two sibling fully connected layer to predict RoI classification and regression. Only 4 channels are applied for each bounding box location because we share the regression between different classes. Benefited from the powerful feature maps for RoI warping, a simple Light-Head R-CNN can also achieve remarkable results, while keeping the efficiency.

RPN (Region Proposal Network) is a sliding-window class-agnostic object detector that use features from C_4. RPN pre-defines a set of anchors, which are controlled by several specific scales and aspect ratios. In our models, we set three aspect ratios $\{1{:}2,\ 1{:}1,\ 2{:}1\}$ and five scales $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ to cover objects of different shapes. Since there are many proposals heavily overlapping with each other, non-maximum suppression (NMS) is used to reduce the number of proposals. Before feeding them into RoI prediction subnetwork. We set the intersection-over-union (IoU) threshold of 0.7 for NMS. Then we assign anchors training labels based on their IoU ratios with ground-truth bounding boxes. If the anchor has IoU over 0.7 with any ground-truth box, it will be set a positive label. Anchors which have highest IoU for ground-truth box will also be assigned a positive label. Meanwhile, if extra anchors have IoU less than 0.3 with all ground-truth box, their labels will be negative. More details can be referred to [28].

4. Experiments

In this section, we evaluate our approach on COCO [21, 18] dataset, which has 80 object categories. There are 80k *train* set and 40k *validation* set, which will be further split into 35k *val-minusmini* and 5k *mini-validation* set. Following the common setting, we combine the *train* set and the *val-minusmini* to obtain the 115K images for training use the 5K *mini-validation* images for validation.

4.1. Implementation Details

Our detector is end-to-end trained based on 8 Pascal TITAN XP GPUs using synchronized SGD with a weight decay of 0.0001 and momentum of 0.9. Each mini-batch has 2 images per GPU and each image has 2000/1000 RoIs for training/testing. We pad images within mini-batch to the same size by filling zeros into the right-bottom of the image. Learning rate is set to 0.01 for first 1.5M iterations (passing 1 image will be regarded as 1 iteration), and 0.001 for later 0.5M iterations.

All experiments adopt *à*trous [24, 23, 2] algorithm in stage 5 of Resnet and online hard example mining (OHEM) [31] techniques. Unless explicitly stated, our backbone network is initialized based on the pre-trained ImageNet [30] base model and pooling size is set to 7. We fix the parameters of stage 1 and 2 in base model, batch normalization is also fixed for faster experiment. Horizontal image

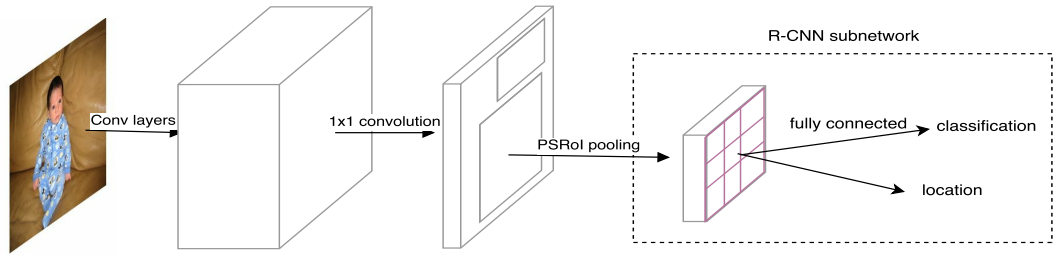

Figure 4. The network to evaluate the impact of thin feature maps. We keep the networks same as R-FCN except that we decrease the feature map channels used for PSRoI pooling. And we add additional fully-connected layers for final prediction.

flipping augmentation is adopted unless otherwise noted.

In the following discussion, we will first perform a series of ablation experiments to validate the effectiveness of our method. Later, we will present the comparison with the state-of-art detectors, such as FPN [19], Mask R-CNN[7], RetinaNet[20], on the COCO *test* dataset.

4.2. Ablation Experiments

In order to fairly compare with the existing methods, we adopt Resnet101 as our backbone network for our ablation studies, similar to the Setting L described in Section3.2.

4.2.1 Baselines

Following the detail setting provided by the publicly available R-FCN code[1], we first evaluate R-FCN in our experiment, denoted as *B1*, and it achieves 32.1% mmAP in COCO *mini-validation* set.

By revising the base setting, we can obtain a stronger baseline, denoted as *B2*, with the following differences: (i) We resize the shorter edge of image to 800 pixels, and restrict the max size of the longer edge to 1200. We set 5 anchors $\{32^2, 64^2, 128^2, 256^2, 512^2\}$ for RPN because of the larger image. (ii) We find that regression loss is definitely smaller that classification loss in R-CNN. Thus we double the R-CNN regression loss to balance multi-task training. (iii) We select 256 samples ranked based on the loss for backpropagation. We use 2000 RoIs per image for training and 1000 RoIs for testing. As table 1 shows, our method improves the mmAP by nearly 3 points.

Models	mAP	AP_s	AP_m	AP_l
B1	32.1	12.8	34.9	46.1
B2	35.2	19.1	39.6	46.5

Table 1. Baselines of our proposed methods. Baseline *B1* is the original R-FCN. Baseline *B2* involves our reproduced implementation details, such as more balanced loss and large image size.

[1] https://github.com/msracver/Deformable-ConvNets

4.2.2 Thin feature maps for RoI warping

We investigate the impact of reducing channels of feature maps for ROI warping following the baseline setting described in Section 4.2.1. To achieve this goal, we design a simple network structure which is shown in Figure 4. The whole pipeline is exactly same as original R-FCN for fairly comparison. Except the following small difference: (i) We reduce the feature map channels to 490 ($10 \times 7 \times 7$) for PSRoI pooling. Noticing it is quite different from the original R-FCN, which involves 3969 ($81 \times 7 \times 7$) channels. (ii) As we modify the channel number of feature maps, we can't directly vote for the final predictions. A simple fully connected layer is involved for final prediction.

Results are shown in table 2. Although the channel number has been largely decrease, 3969 vs 490, our have comparable performance. In addition, it is important to note that, with our Light-head R-CNN design, it enables us to efficiently integrate feature pyramid network [19] as shown in Table 5. This is almost impossible for the original R-FCN as the memory consumption will be high if we want to perform position-sensitive pooling on the high-resolution feature maps like Conv2 (Resnet stage 2).

To validate the influence of reducing channels in Faster R-CNN, we also try to replace the PSRoI pooling with conventional RoI pooling, it slightly improves the accuracy with 0.3 gain. One hypothesis is that RoI pooling involves more features (49x) in second stage, the accuracy gain is benefited from more computation.

Models	mmAP	AP_s	AP_m	AP_l
B1	32.1	12.8	34.9	46.1
thin feature maps *B1*	31.4	12.2	35.5	47.4
B2	35.2	19.1	39.6	46.5
thin feature maps *B2*	34.6	17.6	39.1	48.0

Table 2. The impact of reducing feature map channels for RoI warping. We show the comparison between original R-FCN and R-FCN with thin feature maps for RoI warping. Also we show the influence in our stronger baseline. We use mmAP to indicate results of mAP@[0.5:0.95].

Large separable convolution The feature map channels for RoI warping is pretty small in our design. In original implementation, 1×1 convolution is involved to reshape a small channel, which decrease feature map capability. We involve large separable convolution to enhance these feature maps while keeping small channels. We show the structure of large kernel in Figure 3. The hyperparameter we set is $k = 15, C_{mid} = 256, C_{out} = 490$. In table 3, compared with the results based on our reproduced R-FCN setting *B2*, the thin feature map produced by large kernel can improve the performance by 0.7 points.

Models	mmAP	AP_s	AP_m	AP_l
B2	35.2	19.1	39.6	46.5
+Large Kernel	35.9	19.2	40.4	48.3

Table 3. The impact of enhance thin feature map for RoI warping. *B2* is our implemented strong R-FCN baseline.

4.2.3 R-CNN subnet

Here we evaluate the impact of Light version of R-CNN in R-CNN subnet as shown in Figure 3. A single fully connected layer (without dropout) with 2048 channels is employed in RoI subnetwork. Since the feature map we pooled on is small in channels (10 in our experiments), Light R-CNN is extremely fast for per-region classification and regression. The comparison of the light-RCNN along with our reproduced stronger Faster R-CNN and R-FCN (*B2* is shown in Table 4.

Our methods achieve 37.7 mmAP when combined large kernel feature maps and Light RCNN. In our experiments, under same basic settings, Faster R-CNN and R-FCN get results of 35.5/35.1 mmAP, which is much lower than our methods. More importantly, because of extremely thin feature maps and light R-CNN, we keep the time efficiency even thousands of proposals are utilized.

Models	mAP@[0.5:0.95]
baseline Faster R-CNN	35.6
B2 (R-FCN)	35.2
+Large Kernel	35.9
+Large Kernel and Light-Head R-CNN	37.7

Table 4. The effectivenss of Light-Head R-CNN. The baselines of R-FCN and Fast R-CNN based on our setting L (3.2).

4.3. Light-Head R-CNN: High Accuracy

To combine with state-of-art object detectors, we following the Setting L described in Section 3.2. In addition, we involve interpolation techniques for PSRoI pooling which is proposed in RoIAlign [7]. Results are shown in table 6. It can bring 1.3 points gain. We also apply the common used scale jitter approach. During our training, we randomly sample the scale from {600, 700, 800, 900, 1000} pixels, then resize the shorter edge of the image into the sampled scale. The max edge of the image is constrained within 1400 pixels because the shorter edge may reach to 1000 pixels. Training time is also increased due to the data augmentation. Multi-scale training brings nearly 1 points improvement on mmAP. We finally replace original 0.3 threshold with 0.5 for Non-maximum Suppression (NMS). It improves 0.6 points of mmAP by improving the recall rate especially for the crowd cases.

Table 5 also summarizes the results from state-of-the-art detectors including both one-stage and two-stage methods on COCO test-dev datasets. Our single scale tested model can achieve 40.8% mmAP without bells and whistles, significantly surpassing all the competitors (40.8 vs 39.1). It validates that our Light Head R-CNN would be a good choice for the large backbone models and promising results can be obtained without heavy computation cost. Some of the illustrative results are shown in Figure 5.

4.4. Light-Head R-CNN: High Speed

Larger backbone like Resnet 101 is slow based on the computational speed. To validate the efficiency of Light-Head R-CNN, we produce an efficient bottleneck xception like network with 34.1 top 1 error (at 224×224 in ImageNet) to evaluate our methods. The backbone network architecture is shown in Table 7. Following xception design strategies, we replace all convolution layer in bottle-neck structure with channel-wise convolution. However we do not use the pre-activation design which is proposed in identity mappings [10] because of shallow network. The implementation details for our Light Head R-CNN can be found from the Setting S in Section 3.2.

More specifically, we have the following changes for the fast inference speed: (i) We replace Resnet-101 backbone in Setting L with a tiny xception like network. (ii) We abandon àtrous algorithm in our fast models, because it involves much computation compared with small backbone. (iii) we set RPN convolution to 256 channels, which is half of original used in Faster R-CNN and R-FCN. (iv) we apply large separable convolution with $kernel\_size = 15, C_{mid} = 64, C_{out} = 490$ ($10 \times 7 \times 7$). Since the middle channels we used is extremely small, large kernel is still efficient for inference. (v) We adopt PSPooling with alignment technique as our RoI warping, since it reduces the pooled feature map channels by $k \times k$ times (k is the pooling size). Noticing that it will obtain better results if we involve RoI-align.

We compared our methods with recent fast detectors like YOLO, SSD, and DeNet. Results are evaluated on COCO test-dev datasets, only one batch is adopted and we absorb

method	test size shorter edge/max size	feature pyramid	align	mAP@[0.5:0.95]	AP_s	AP_m	AP_l
R-FCN [17]	600/1000			32.1	12.8	34.9	46.1
Faster R-CNN (2fc)	600/1000			30.3	9.9	32.2	47.4
Deformable [3]	600/1000		√	34.5	14.0	37.7	50.3
G-RMI [13]	600/1000			35.6	-	-	-
FPN [19]	800/1200	√		36.2	18.2	39.0	48.2
Mask R-CNN [7]	800/1200	√	√	38.2	20.1	41.1	50.2
RetinaNet [20]	800/1200	√		37.8	20.2	41.1	49.2
RetinaNet ms-train [20]	800/1200	√		39.1	21.8	42.7	50.2
Light head R-CNN	800/1200		√	**39.5**	21.8	43.0	50.7
Light head R-CNN ms-train	800/1200		√	**40.8**	22.7	44.3	52.8
Light head R-CNN	800/1200	√	√	**41.5**	25.2	45.3	53.1

Table 5. Comparisons of single size tested single-model results on COCO test-dev. All experiments are using Resnet-101 as basic feature extractor (except R-RMI with Inception Resnet V2 [33]). Light-Head R-CNN reaches a new state-of-the-art accuracy. Noticing results of test-dev is slightly different from mini-validation. "ms-train" means multi-scale training.

Models	mAP@[0.5:0.95]
Light-Head R-CNN	37.7
+ pool with alignment	39.0
+ NMS with 0.5 threshold	39.6
+ multiscale train	40.6

Table 6. Further improvements for our approach. Results are evaluated on COCO mini-validation set.

Layer	Output size	KSize	S	Rep	output channels
Image	224×224				
Conv1	112×112	3×3	2	1	24
MaxPool	56×56	3×3	2		
Stage2	28×28		2	1	144
	28×28		1	3	144
Stage3	14×14		2	1	288
	14×14		1	7	288
Stage4	7×7		2	1	576
	7×7		1	3	576
GAP	1×1	7×7			
FC					1000
Comp*					145M

Table 7. An efficient Xception like architecture for our fast detector. Comp* indicates the complexity of the networks (FLOPs)

the Batch-normalization [15] for fast inference. As shown in Table 8, our methods get 30.7 mmAP at 102 FPS on MS COCO, significantly outperforming the fast detectors like YOLO and SSD. Some of the results are visualized in Figure 6.

5. Conclusion

In this paper, we present Light Head R-CNN, which involves a better design principle for two-stage object detectors. Compared with traditional two-stage detectors like Faster R-CNN and R-FCN, which usually have a heavy head, our light head design enables us to significantly improve the detection results, without compromising the computational speed. More importantly, compared with the fast single-stage detectors like YOLO and SSD, we obtain superior performance even with faster computational speed. For example, our Light Head R-CNN coupled with small Xception-like base model can achieve 30.7 mmAP at the speed of 102 FPS.

Model	backbone	test size short/max edge	speed (fps)	mAP @[0.5:0.95]
YOLO V2	Darknet	448/448	40	21.6
SSD	VGG	300/300	58	25.1
SSD	Resnet101	300/300	16	28.0
SSD	Resnet101	500/500	8	31.2
DSSD [4]	Resnet101	300/300	8	28.0
DSSD	Resnet101	500/500	6	33.2
R-FCN	Resnet101	600/1000	11	29.9
DeNet	Resnet34	512/512	83	29.4
Light Head R-CNN	xception*	800/1200	**95**	**31.5**
Light Head R-CNN	xception*	700/1100	**102**	**30.7**

Table 8. Comparisons of our fast detector results on COCO test-dev. Xception* is a small xception like backbone. By involving a tiny base-model, Light R-CNN achieves superior performance on both accuracy and speed, which shows the flexiblilty of our proposed design.

References

[1] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 328–335, 2014.

Figure 5. Representative results of our large "L" model.

Figure 6. Representative results of our small "S" model.

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[3] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *arXiv preprint arXiv:1703.06211*, 2017.

[4] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. *arXiv preprint arXiv:1701.06659*, 2017.

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

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *arXiv preprint arXiv:1703.06870*, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[13] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. *arXiv preprint arXiv:1611.10012*, 2016.

[14] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[15] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[17] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016.

[18] T.-Y. Lin and P. Dollár. Ms coco api. https://github.com/pdollar/coco. 2016.

[19] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016.

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*, 2017.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[24] S. Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

[25] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters–improve semantic segmentation by global convolutional network. *arXiv preprint arXiv:1703.02719*, 2017.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[27] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[29] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1476–1481, 2017.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[31] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[33] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[35] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[36] L. Tychsen-Smith and L. Petersson. Denet: Scalable real-time object detection with directed sparse sampling. *arXiv preprint arXiv:1703.10295*, 2017.

[37] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.

[38] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995. IEEE, 2017.

[39] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.

[40] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision*, pages 391–405. Springer, 2014.