Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

2021-02-09

전세희

E-DEEP

Abstract

In this work, we introduce a ***Region Proposal Network*** **(RPN)** that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals.

The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection

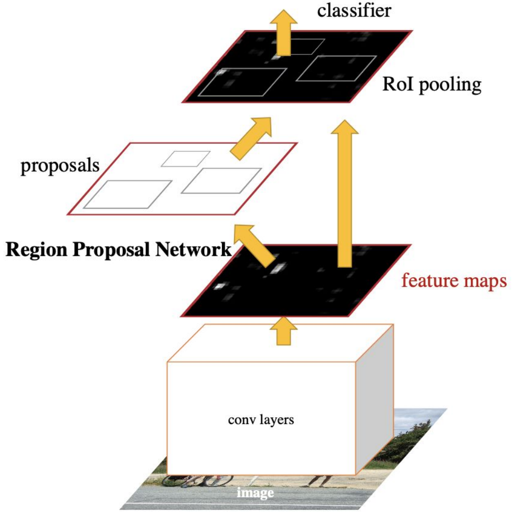


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

Introduction

Recent advances in object detection are driven by the success of **region proposal methods** (*e.g.*, [4]) and **region-based convolutional neural networks** (R- CNNs) [5]

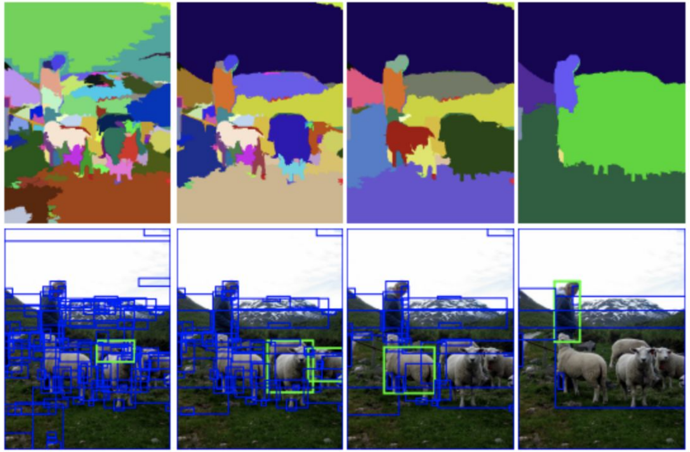
The latest incarnation, Fast R-CNN [2], achieves near real-time rates using very deep networks [3], ***when ignoring the time spent on region proposals*.**

👉🏻 많은 region proposal 방법(ex. Selective Search, EdgeBoxes)들이 제안되었지만
여전히 너무 시간이 오래걸린다.!!!!

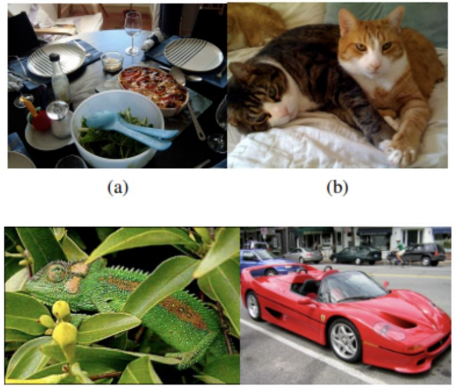
👉🏻 In this paper, we show an elegant and effective solution where proposal computation is **nearly cost-free given the detection network's computation.**

Selective Search

- **Capture All Scales** : 이미지 내에서 물체의 크기는 랜덤하다. 또한 일부 객체는 다른 객체보다 경계가 덜 명확하다. 따라서, 모든 객체의 크기를 고려해야 한다.



- **Diversification** : 영역들을 그룹화하는데 있어 최적화된 단일 전략은 없다. 따라서 색상, 재질(텍스쳐), 크기 등 다양한 종류의 조건을 고려하여 다룰 필요가 있다.



(a) (b) (c) (d)

(a) 다양한 크기의 물체, (b) 재질은 유사하지만 색상이 다른 고양이,
(c) 색상이 동일하지만, 재질이 다른 카멜레온, (d) 색상과 재질이 유사하지 않지만 차량의 일부인 바퀴

- **Fast to Compute** : 이 방법의 목표는 실제 객체 탐지 프레임워크에서 사용할 수 있어야 하며, 계산상 병목현상(bottleneck)이 발생하면 안되므로 빨라야 한다.

Selective Search is an order of magnitude slower, at 2 seconds per image in a CPU implementation.

https://go-hard.tistory.com/33

The convolutional feature maps used by region-based detectors, like Fast R- CNN, can also be used for generating region proposals.

On top of these convolutional features, we construct an **RPN** by adding a few **additional convolutional layers** that simultaneously regress region **bounds** and objectness **scores** at each location on a regular grid.

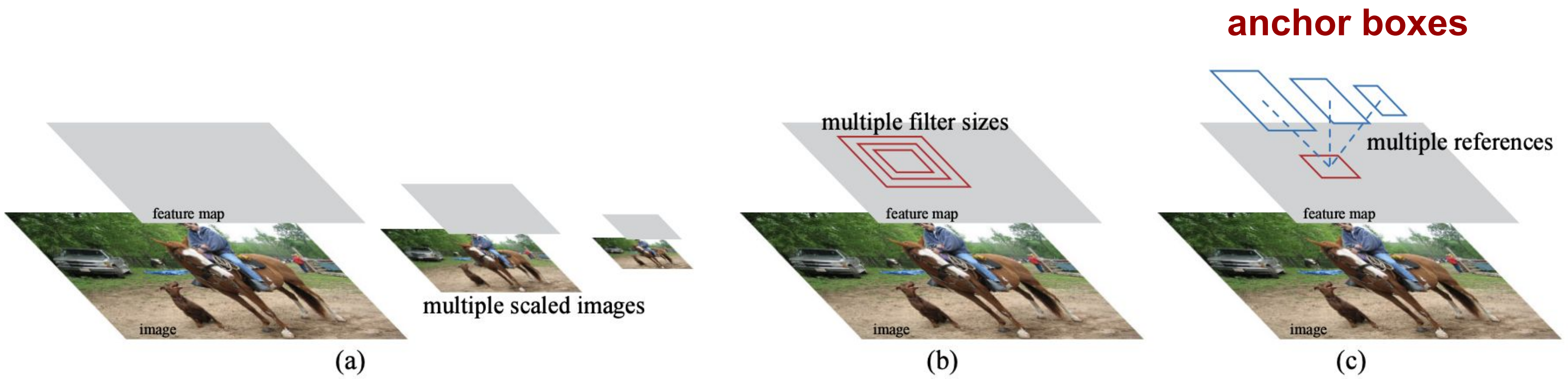


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

Faster RCNN = RPN + Fast RCNN

We propose a training scheme that **alternates** between fine-tuning for **the region proposal task** and then fine-tuning for **object detection**, while keeping the proposals fixed.

```
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
print 'Stage 1 RPN, generate proposals'
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'

mp_kwargs = dict(
        queue=mp_queue,
        imdb_name=args.imdb_name,
        rpn_model_path=str(rpn_stage1_out['model_path']),
        cfg=cfg,
        rpn_test_prototxt=rpn_test_prototxt)
p = mp.Process(target=rpn_generate, kwargs=mp_kwargs)
p.start()
rpn_stage1_out['proposal_path'] = mp_queue.get()['proposal_path']
p.join()
```

```
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
print 'Stage 1 Fast R-CNN using RPN proposals, init from ImageNet model'
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'

cfg.TRAIN.SNAPSHOT_INFIX = 'stage1'
mp_kwargs = dict(
        queue=mp_queue,
        imdb_name=args.imdb_name,
        init_model=args.pretrained_model,
        solver=solvers[1],
        max_iters=max_iters[1],
        cfg=cfg,
        rpn_file=rpn_stage1_out['proposal_path'])
p = mp.Process(target=train_fast_rcnn, kwargs=mp_kwargs)
p.start()
fast_rcnn_stage1_out = mp_queue.get()
p.join()
```

```
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'
print 'Stage 2 RPN, init from stage 1 Fast R-CNN model'
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~'

cfg.TRAIN.SNAPSHOT_INFIX = 'stage2'
mp_kwargs = dict(
        queue=mp_queue,
        imdb_name=args.imdb_name,
        init_model=str(fast_rcnn_stage1_out['model_path']),
        solver=solvers[2],
        max_iters=max_iters[2],
        cfg=cfg)
p = mp.Process(target=train_rpn, kwargs=mp_kwargs)
p.start()
rpn_stage2_out = mp_queue.get()
p.join()
```

번갈아가며 학습

```
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
print 'Stage 2 RPN, generate proposals'
print '~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

mp_kwargs = dict(
        queue=mp_queue,
        imdb_name=args.imdb_name,
        rpn_model_path=str(rpn_stage2_out['model_path']),
        cfg=cfg,
        rpn_test_prototxt=rpn_test_prototxt)
p = mp.Process(target=rpn_generate, kwargs=mp_kwargs)
p.start()
rpn_stage2_out['proposal_path'] = mp_queue.get()['proposal_path']
p.join()
```

https://github.com/rbgirshick/py-faster-rcnn/blob/master/tools/train_faster_rcnn_alt_opt.py

✓ Faster R-CNN

Faster R-CNN is composed of two modules.

1. **Deep fully convolutional network that proposed regions**
2. **Fast R-CNN detector that uses the proposed regions**

The RPN module tells the Fast R-CNN module

where to look (like *"attention"*)

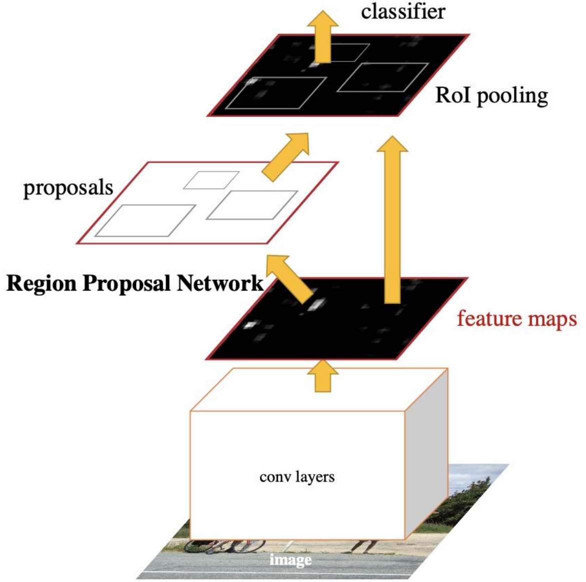
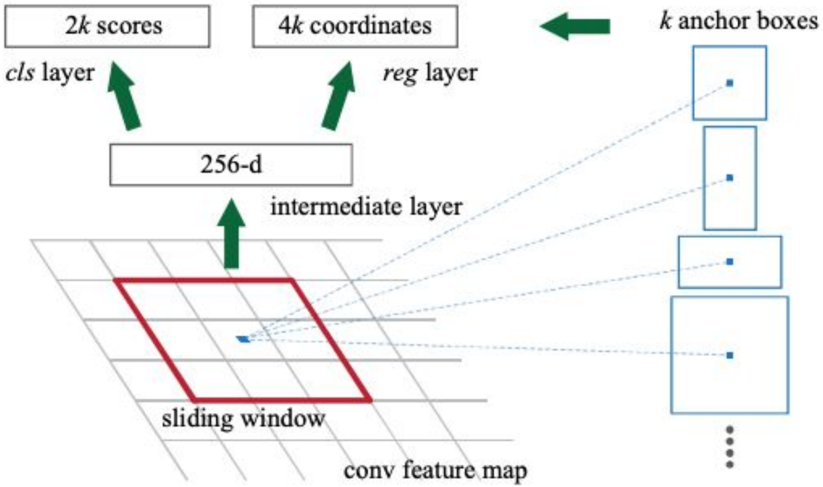


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

1> Region Proposal Networks (RPN)

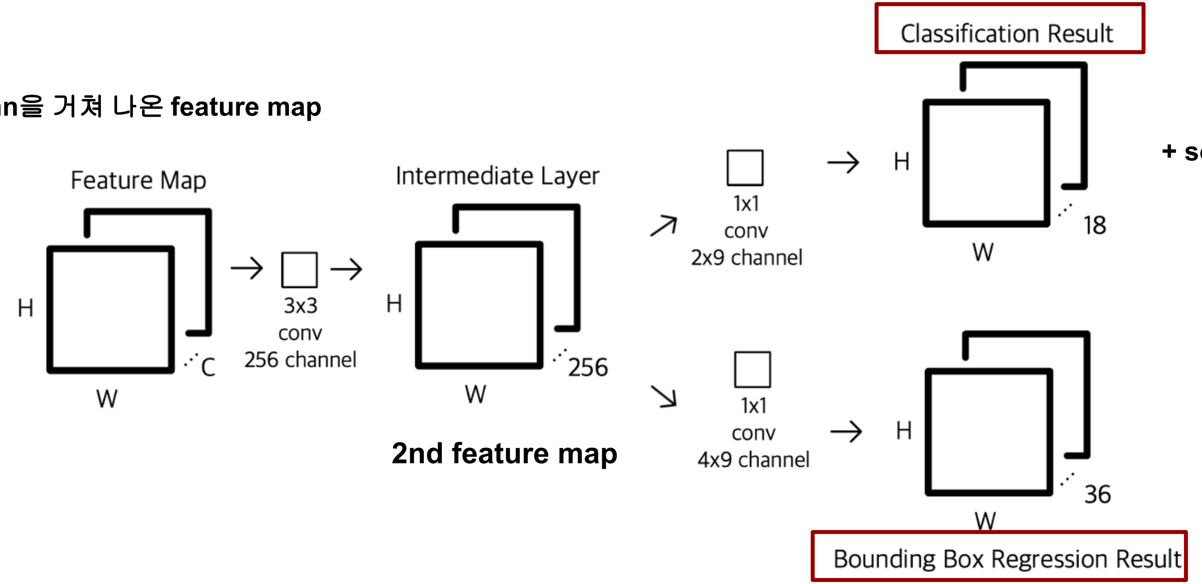
A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of **rectangular object proposals**, each with an objectness **score**.

Because our ultimate goal is to share computation with a Fast R-CNN object detection network [2], we assume that both nets **share a common set of convolutional layers.**



1> Region Proposal Networks (RPN)

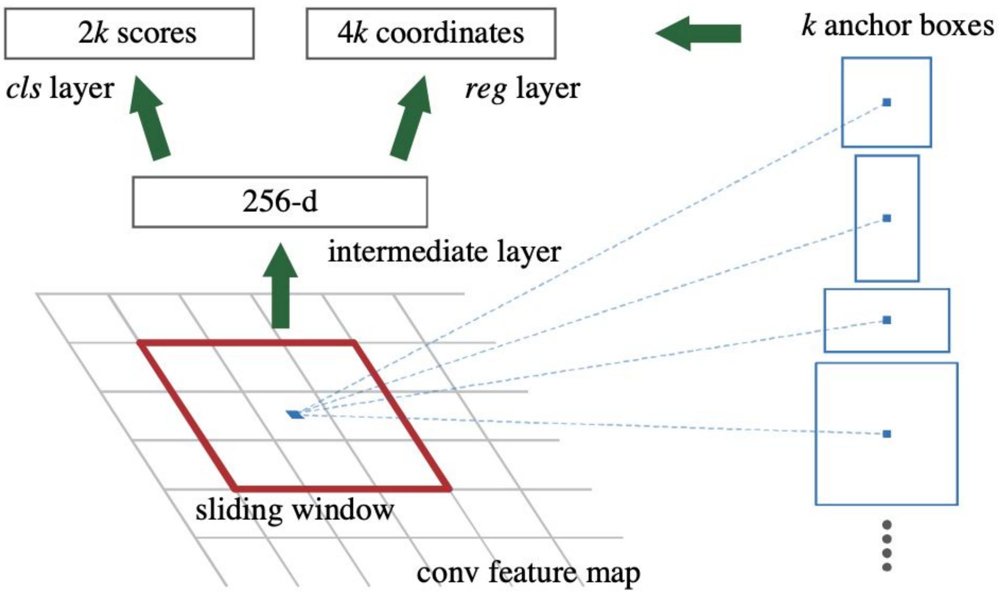
cnn을 거쳐 나온 feature map

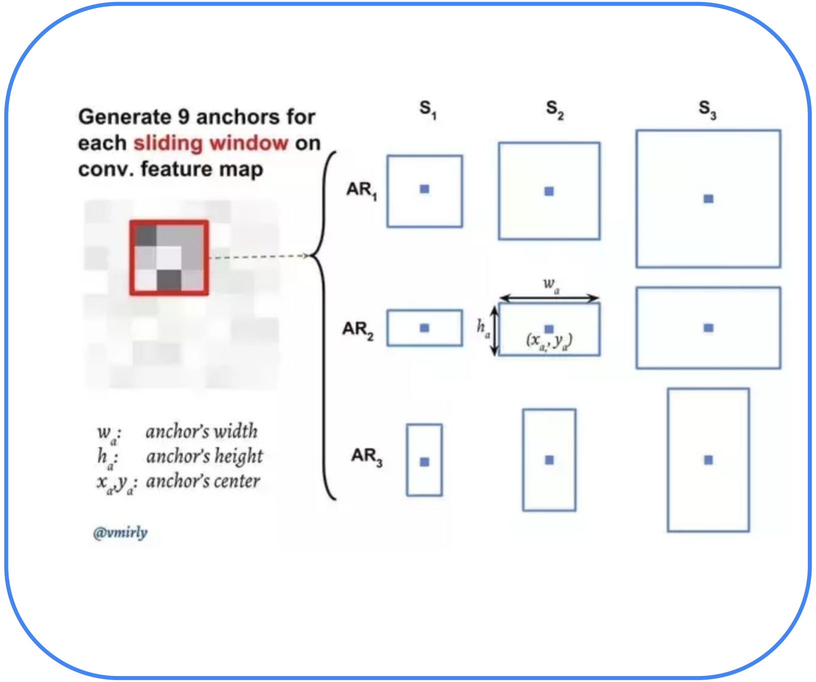


2nd feature map

+ softmax

https://yeomko.tistory.com/17





By default we use 3 scales and 3 aspect ratios, yielding k = 9 anchors at each sliding position.

☝ Anchors : Translation-Invariant

RPN

MultiBox

2.8×10^4 parameters **vs** 6.1×10^6 parameters

(k=9)

👉 This model has less risk of overfitting on small datasets, like PASCAL VOC.

✌️ Multi-Scale Anchors as Regression References

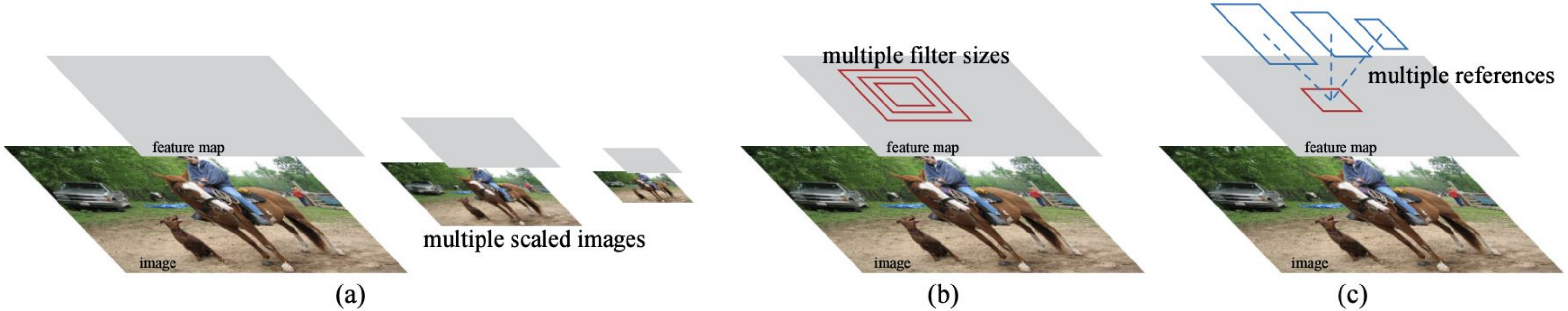
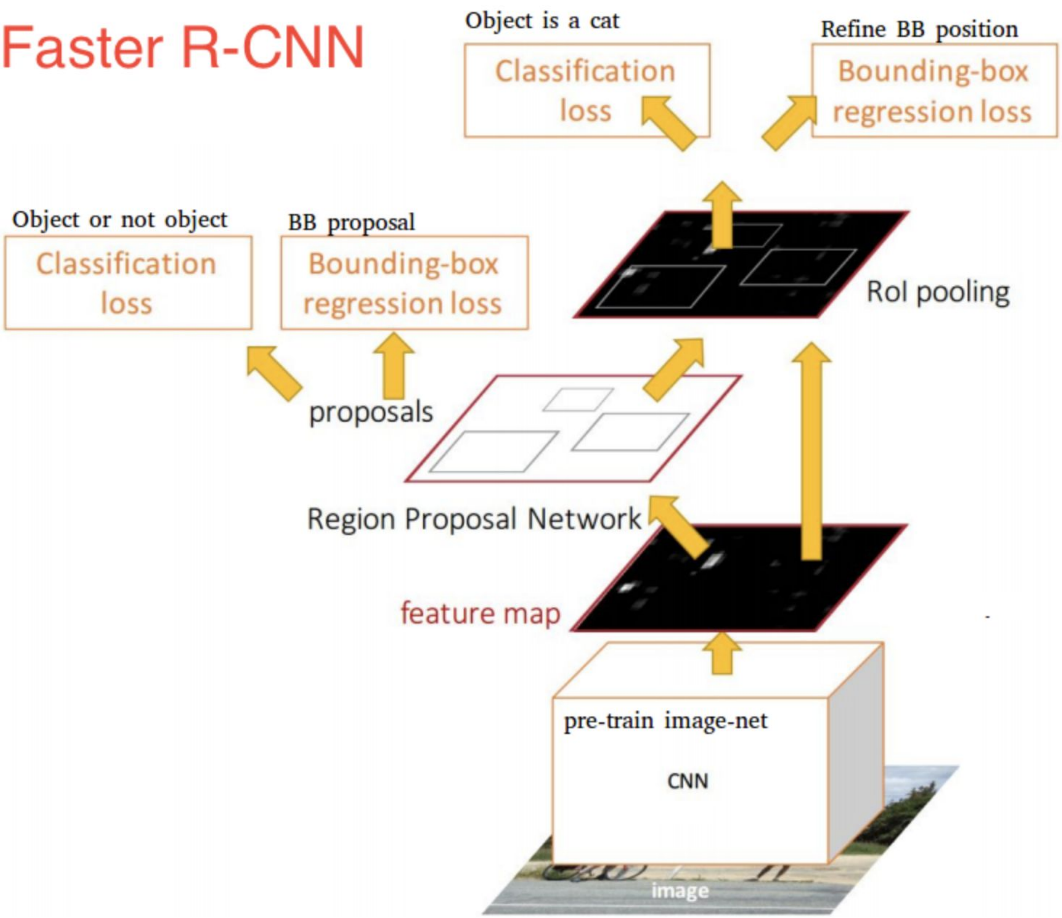


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

Our anchor-based method is built on *a pyramid of anchors*, which is **more cost-efficient.**

The design of multi- scale anchors is a key component for sharing features **without extra cost** for addressing scales.

Faster R-CNN



http://incredible.ai/deep-learning/2018/03/17/Faster-R-CNN/

Loss function

Bounding box regression

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, \boxed{p_i^*}) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

anchor가 positive면 1, 아니면 0

classification

p_i : classification을 통해 얻은 anchor가 object일 확률
t_i : bounding box regression을 통해 얻은 box 좌표값 vector(4-dim)

$$L_{reg}(t_i, t_i^*) = \boxed{R}(t_i - t_i^*) \Rightarrow \text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

Training ✅ 앞에서 언급했듯이 번갈아가며 학습.

<4-Step Alternating Training>

1. ImageNet pre-trained 모델을 이용하여 RPN을 fine-tuning

2. Train a separate detection network by Fast R-CNN using the proposals **generated by the step-1 RPN.** This detection network is also initialized by the **ImageNet-pre-trained model.** (❌The two networks do **not share convolutional layers.)**

3. We use the detector network to initialize RPN training, but we fix the shared convolutional layers and only **fine-tune** the layers.

4. Keeping the shared convolutional layers fixed, we **fine-tune** the unique layers of Fast R-CNN. As such, both networks **share the same convolutional layers** and form a unified network.

Experiments(on PASCAL)

Table 3: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Table 4: Detection results on **PASCAL VOC 2012 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07++12": union set of VOC 2007 trainval+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html. ‡: http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html. §: http://host.robots.ox.ac.uk:8080/anonymous/XEDH10.html.

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Table 6: Results on PASCAL VOC 2007 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000. RPN* denotes the unsharing feature version.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
SS	2000	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
RPN*	300	07	68.5	74.1	77.2	67.7	53.9	51.0	75.1	79.2	78.9	50.7	78.0	61.1	79.1	81.9	72.2	75.9	37.2	71.4	62.5	77.4	66.4
RPN	300	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
RPN	300	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
RPN	300	COCO+07+12	**78.8**	**84.3**	**82.0**	**77.7**	**68.9**	**65.7**	**88.1**	**88.4**	**88.9**	**63.6**	**86.3**	**70.8**	**85.9**	**87.6**	**80.1**	**82.3**	**53.6**	**80.4**	**75.8**	**86.6**	**78.9**

Table 7: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
SS	2000	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	**65.7**	80.4	64.2
RPN	300	12	67.0	82.3	76.4	71.0	48.4	45.2	72.1	72.3	87.3	42.2	73.7	50.0	86.8	78.7	78.4	77.4	34.5	70.1	57.1	77.1	58.9
RPN	300	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
RPN	300	COCO+07++12	**75.9**	**87.4**	**83.6**	**76.8**	**62.9**	**59.6**	**81.9**	**82.0**	**91.3**	**54.9**	**82.6**	**59.0**	**89.0**	**85.5**	**84.7**	**84.1**	**52.2**	**78.9**	65.5	**85.4**	**70.2**

Table 8: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different settings of anchors**. The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using 3 scales and 3 aspect ratios (69.9%) is the same as that in Table 3.

settings	anchor scales	aspect ratios	mAP (%)
1 scale, 1 ratio	128^2	1:1	65.8
	256^2	1:1	66.7
1 scale, 3 ratios	128^2	{2:1, 1:1, 1:2}	68.8
	256^2	{2:1, 1:1, 1:2}	67.9
3 scales, 1 ratio	$\{128^2, 256^2, 512^2\}$	1:1	**69.8**
3 scales, 3 ratios	$\{128^2, 256^2, 512^2\}$	{2:1, 1:1, 1:2}	**69.9**

Table 9: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different values of** λ in Equation (1). The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using $\lambda = 10$ (69.9%) is the same as that in Table 3.

λ	0.1	1	10	100
mAP (%)	67.2	68.9	69.9	69.1

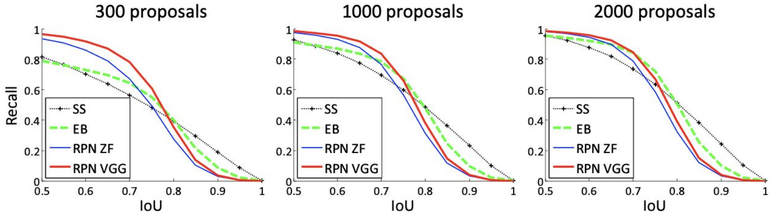


Figure 4: Recall *vs*. IoU overlap ratio on the PASCAL VOC 2007 test set.

Table 10: **One-Stage Detection *vs*. Two-Stage Proposal + Detection**. Detection results are on the PASCAL VOC 2007 test set using the ZF model and Fast R-CNN. RPN uses unshared features.

	proposals		detector	mAP (%)
Two-Stage	RPN + ZF, unshared	300	Fast R-CNN + ZF, 1 scale	58.7
One-Stage	dense, 3 scales, 3 aspect ratios	20000	Fast R-CNN + ZF, 1 scale	53.8
One-Stage	dense, 3 scales, 3 aspect ratios	20000	Fast R-CNN + ZF, 5 scales	53.9

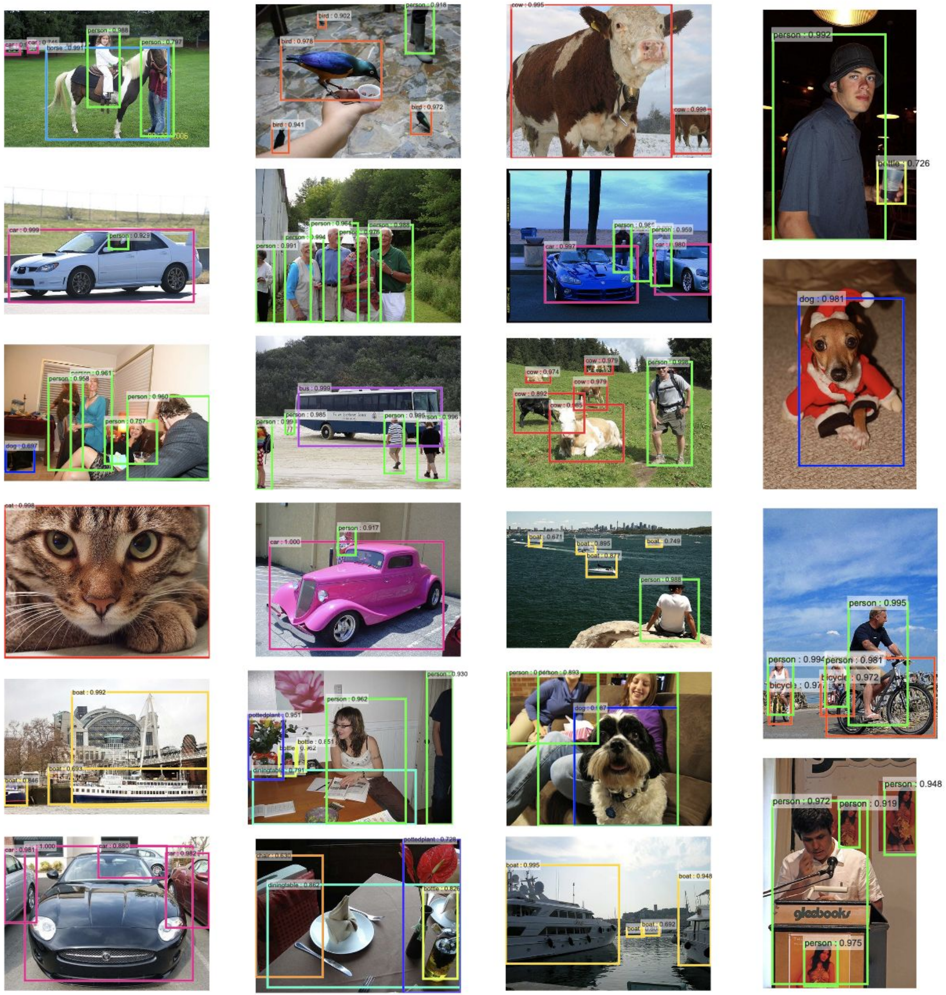
Experiments(on COCO)

Table 11: Object detection results (%) on the **MS COCO** dataset. The model is VGG-16.

method	proposals	training data	COCO val		COCO test-dev	
			mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

Table 12: Detection mAP (%) of Faster R-CNN on PASCAL VOC 2007 test set and 2012 test set using different training data. The model is VGG-16. "COCO" denotes that the COCO trainval set is used for training. See also Table 6 and Table 7.

training data	2007 test	2012 test
VOC07	69.9	67.0
VOC07+12	73.2	-
VOC07++12	-	70.4
COCO (no VOC)	76.1	73.0
COCO+VOC07+12	**78.8**	-
COCO+VOC07++12	-	**75.9**


person : 0.992
bottle : 0.726
dog : 0.981
person : 0.995
person : 0.981
bicycle : 0.972
person : 0.972
person : 0.919
person : 0.948
person : 0.975