Paper-Review

이 준 걸

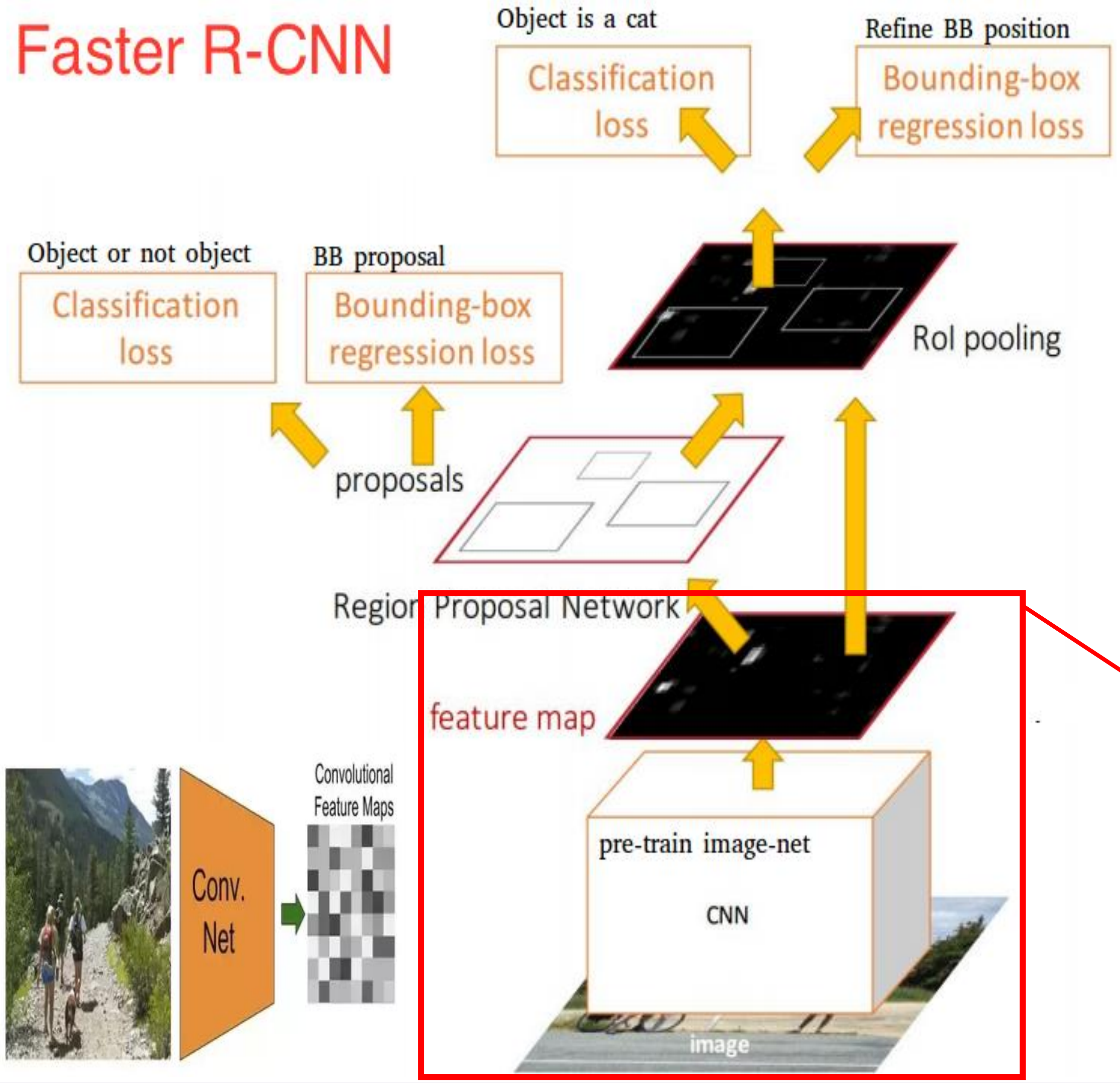
Faster R-CNN

Why Faster R-CNN? - Introduction

1. "Fast R-CNN achieves near real-time rates using very deep networks, when ignoring the time spent on region proposals."

2. "Selective Search(SS) is an order of magnitude slower, at 2s per image in a CPU implementation."
-> Exactly Real-time analysis is impossible

3. "We introduce novel *Region Proposal Networks(RPNs)* that share convolutional layers with state-of-art object detection networks."
-> "The marginal cost for proposals is small(10ms per image)" ; Real-time analysis is possible



Architecture

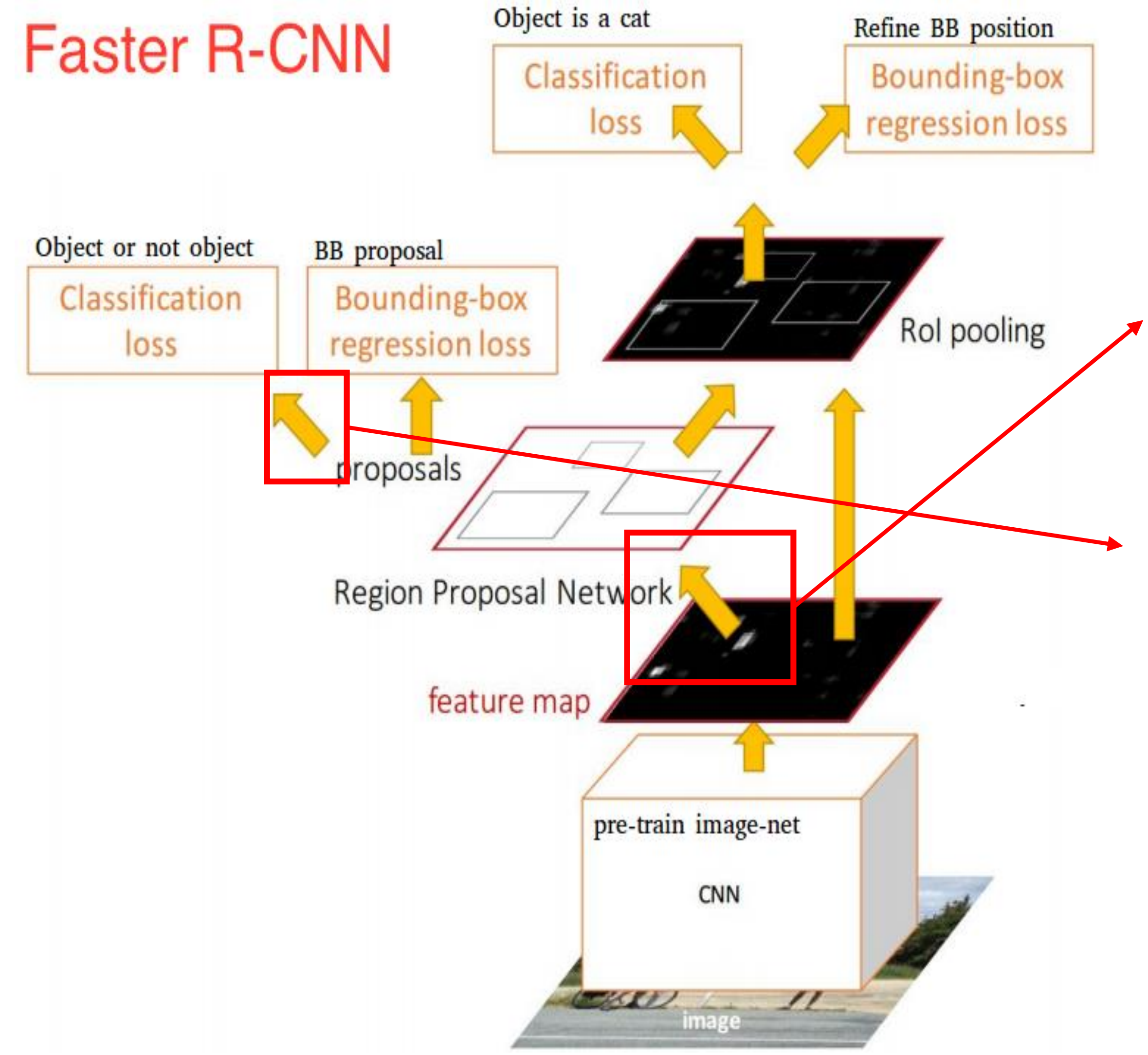
- Input Images : Height x Width X Depth(RGB)

- 이전 R-CNN에서는 SS를 통해 나온 수천 개의 Region을 Alexnet으로 분류.

- 또한 3개의 모델(Feature를 뽑아내는 CNN, 어떤 Class인지 알아내는 Classifier, Bounding Boxes 예측하는 Regression model)을 각각 학습해야 했음.

- Fast R-CNN에서는 중복되는 연산을 하나의 CNN으로 해결.

- 즉, 이미지를 가장 먼저 받아서 Feature를 뽑아내는 역할을 CNN에서 처리하기 때문에 Base Network라고 함.



Architecture

We construct RPNs by adding two additional conv layer:

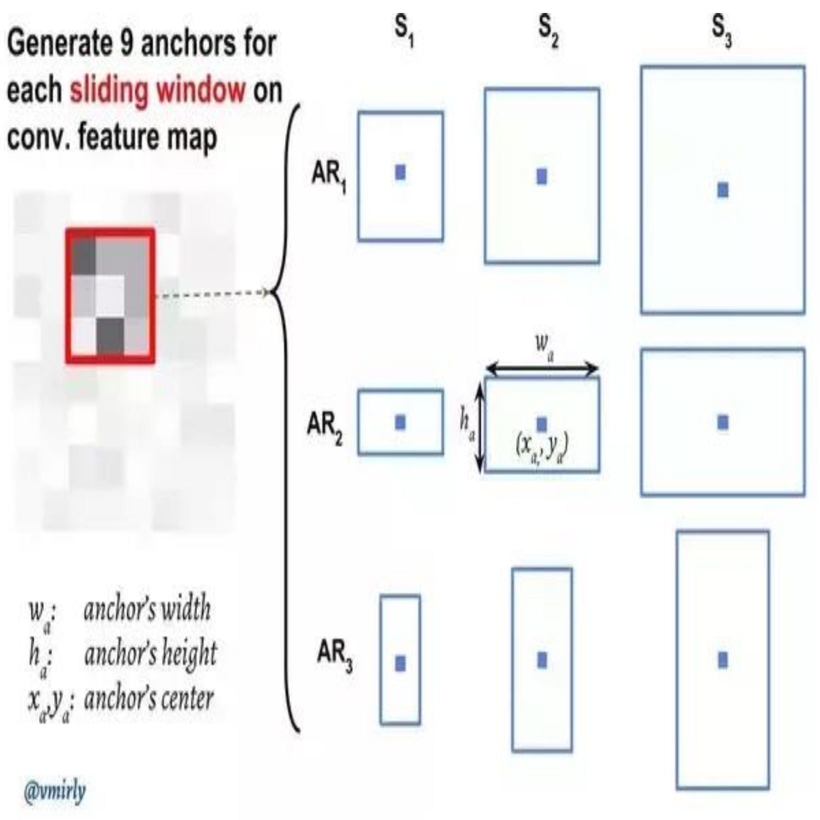
1. Region Proposal Networks(RPNs) : Deep Convolution Network

"They(RPNs) can be trained end-to-end specifically for the task for generating detection proposals."

2. Fast R-CNN Detector : Use Proposed Regions that suggested by RPNs and Classify Object

Faster R-CNN안에는 2개의 모듈이 존재하지만, 전체적으로는 하나의 object detection network라고 볼 수 있음.

Region Proposal Networks

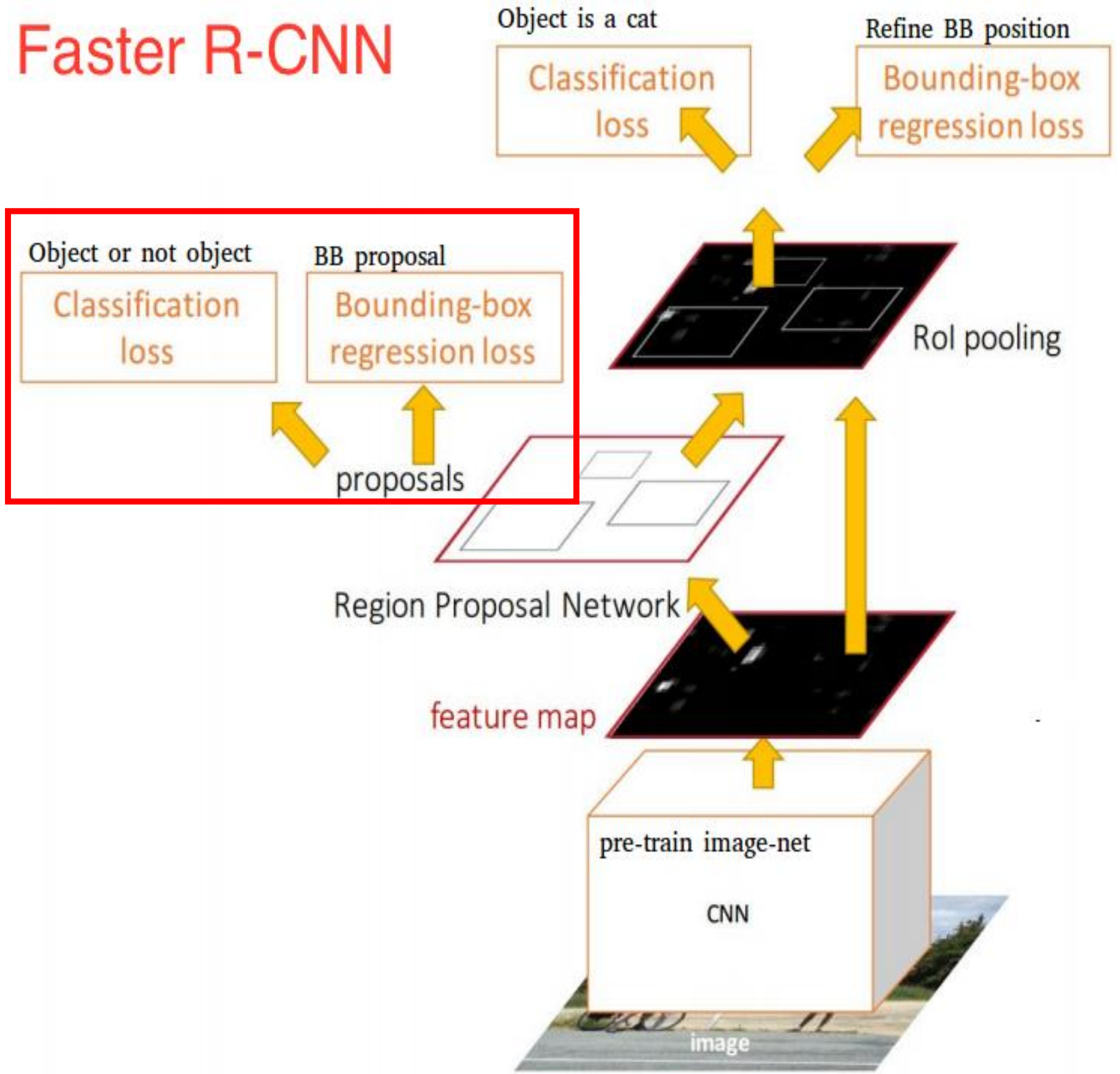


1. "At each Sliding-Window location, we simultaneously predict k region"

2. N*N(보통 3 X 3) spatial window를 슬라이드 시킴. 각 지점에서 한번에 여러 개의 Region Proposals을 예측하게 됨. Region Proposals 개수를 k로 나타내며 이것을 Anchor라고 부름

3. "We use 3 scales and 3 aspect ratios, yielding k=9 anchors at each sliding position. "

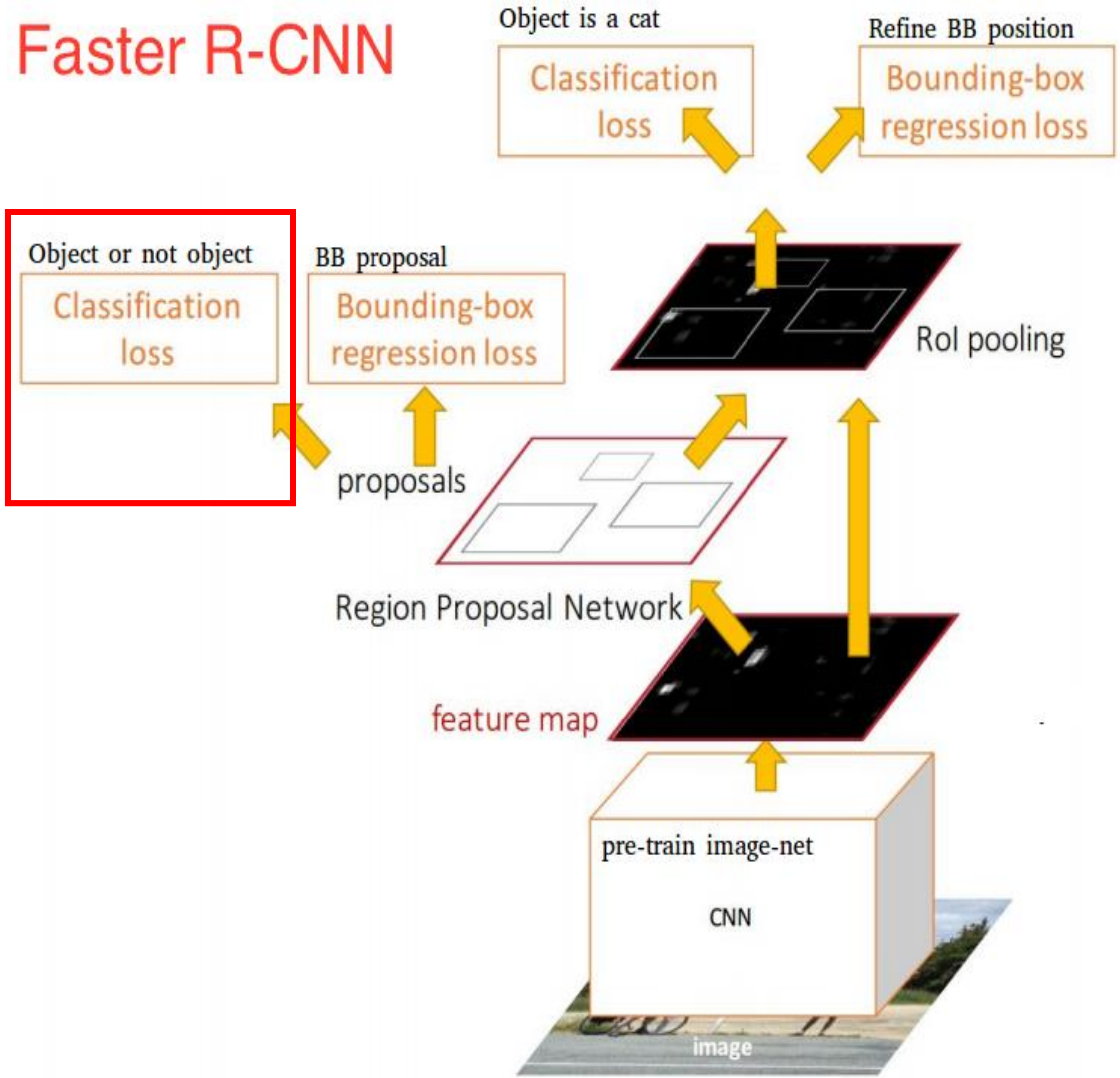
4. "An important property of our approach is that it is translation invariant, both in terms of the anchors and the functions that compute proposals relative to the anchors"



The fully-connected layers are shared across all spatial locations.

Cls layer : 1X1 filter with 1 stride and 0 padding을 9*2(=18)개 적용하여 14X14X9X2의 이웃풋을 얻는다. 여기서 filter의 개수는, anchor box의 개수(9개) * score의 개수(2개: object / non-object)로 결정된다.

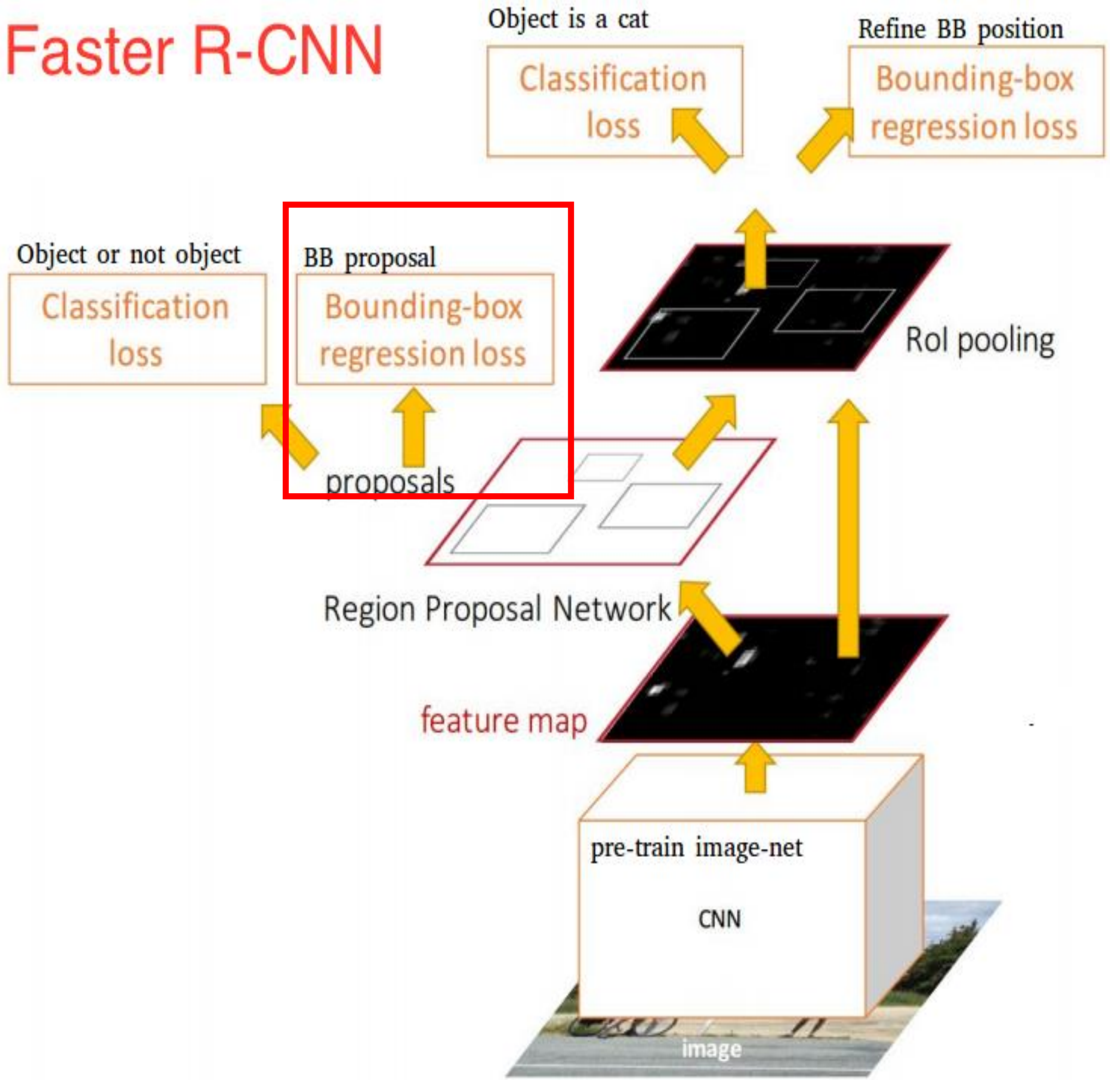
reg layer: 1X1 filter with 1 stride and 0 padding을 9*4(=36)개 적용하여 14X14X9X4의 아웃풋을 얻는다. 여기서 filter의 개수는, anchor box의 개수(9개)



Classifier of Background and Foreground

- **Classifier**를 학습시키기 위한 training data는 바로 위의 **RPN으로부터 얻은 anchors** 와 **ground-truth boxes** (실제 사람이 집접 박스 처리한 데이터)
- 모든 anchors를 foreground 이냐 또는 background이냐로 분류를 해야함
- 각각의 anchor마다 foreground인지 아니면 background인지 구별하는 값을 p* 값이라고 했을 때 구체적인 공식은 다음과 같음

$$p^* = \begin{cases} 1 & \text{if } IoU > 0.7 \\ -1 & \text{if } IoU < 0.3 \\ 0 & \text{if otherwise} \end{cases} \qquad IoU = \frac{\text{anchor} \cap \text{ground-truth box}}{\text{anchor} \cup \text{ground-truth box}}$$



Bounding Box Regression

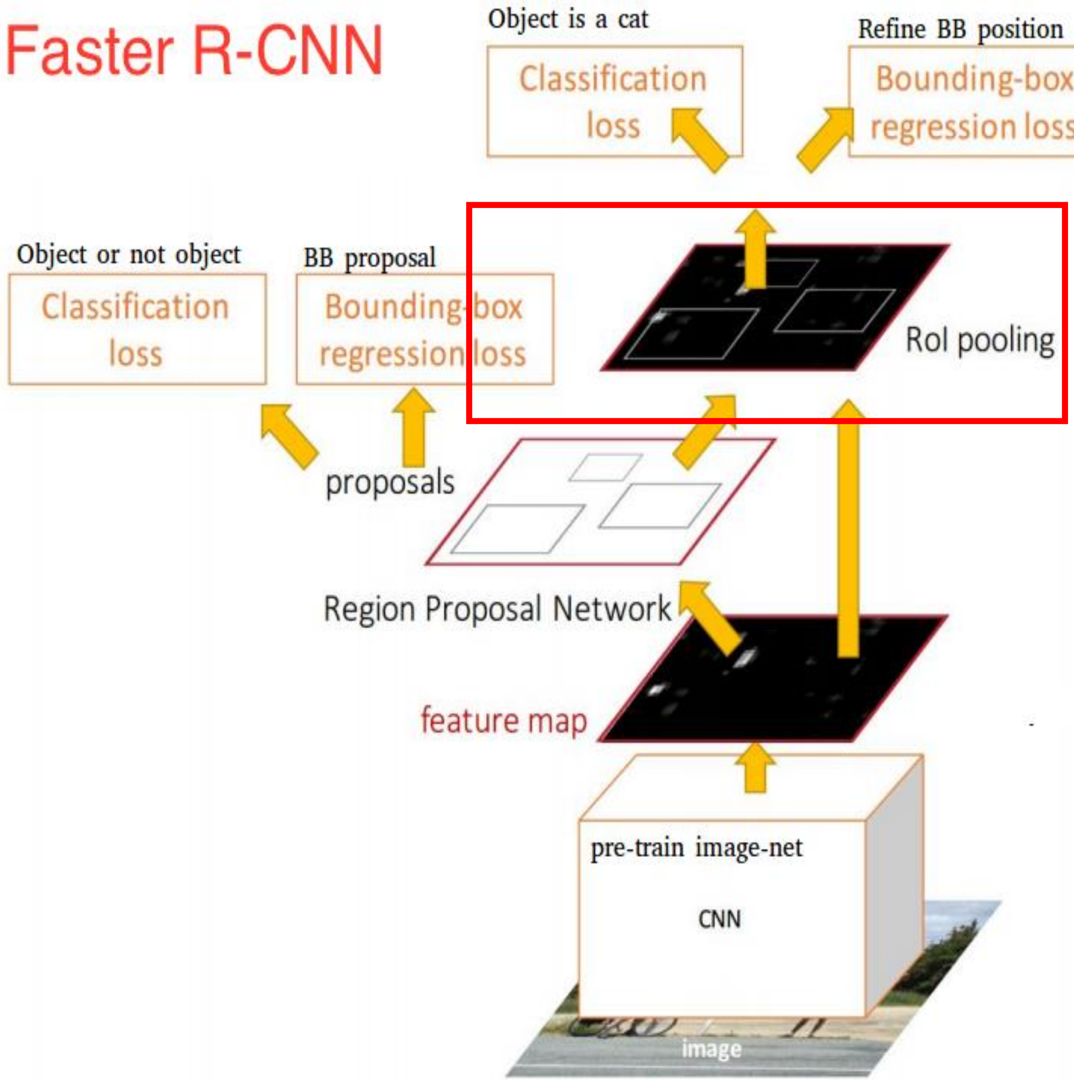
- Bounding box regression에는 4개의 좌표값을 사용함. t라는 값 자체가 4개의 좌표값을 갖고 있는 하나의 벡터라고 보면 되며 다음과 같은 엘러먼트 값을 갖고 있음

$$t_x = (x - x_a)/w_a$$
$$t_y = (y - y_a)/h_a$$
$$t_w = \log(w/w_a)$$
$$t_h = \log(h/h_a)$$

- ground-truth vector t 에는 위와 유사하게 다음과 같은 값을 갖고 있습니다.

$$t_x^* = (x^* - x_a)/w_a$$
$$t_y^* = (y^* - y_a)/h_a$$
$$t_w^* = \log(w^*/w_a)$$
$$t_h^* = \log(h^*/h_a)$$

- t_x, t_y : 박스의 center coordinates
- t_w, t_h : 박스의 width, height
- x, y, w, h : predicted box
- x_a, y_a, w_a, h_a : anchor box
- x^*, y^*, w^*, h^* : ground-truth box

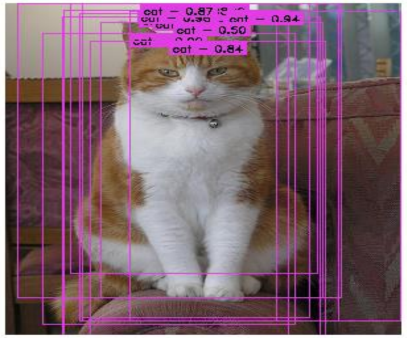


Region of Interest Pooling

- RPN이후, 서로 다른 크기의 Proposed Regions값을 Output으로 받게 되나 이 들의 크기는 제 각각임.

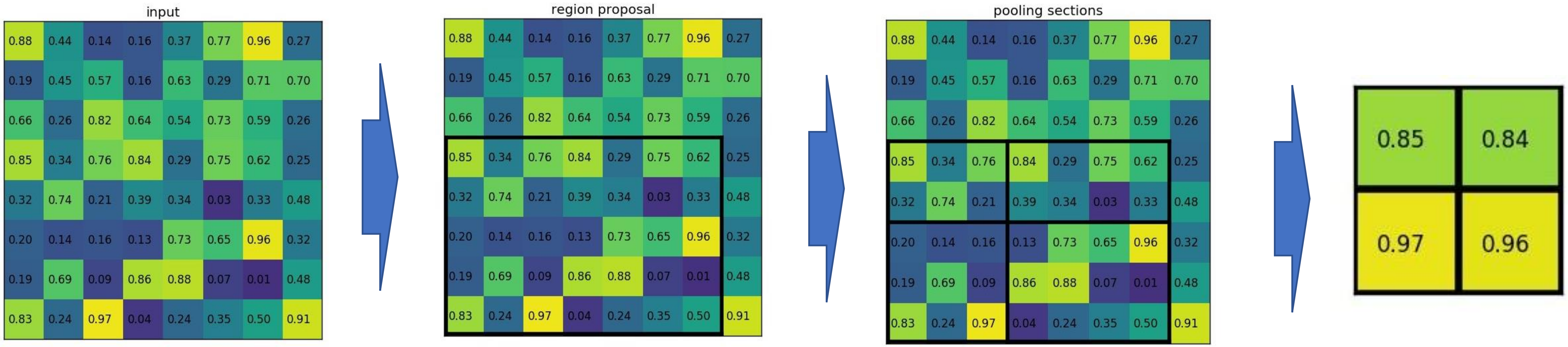
- 따라서 이때 ROI기법을 쓰게 되면 서로 다른 크기의 Feature map 을 동일한 크기로 변환됨.

- ROI구현하기 위해서는 2개의 Input이 필요함
  1. Deep Convoltions 그리고 Max pooling Layers통해 나온 Feature map
  2. N * 4 Matrix -> N은 ROI의 개수, 4는 Region의 위치를 나타내는 좌표



Region of Interest Pooling

- 로직은 다음과 같음.
    1. 각각의 Region Proposal을 동일한 크기의 Section으로 나눔(Section의 크기는 ROI Pooling의 Output크기가 동일함)
    2. 각각의 Section에서 가장 큰값을 찾음(a.k.a Max Pooling)
    3. 각각의 찾은 Max값을 Output으로 뱉음(a.k.a Max Pooling)



Loss Function

i = anchor index in minibatch

Ground truth objectness label

Log loss

Smooth L1 loss

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

True box coordinates

Coordinates of the predicted bounding box for anchor *i*

Predicted probability of being an object for anchor *i*

In practice λ= 10, so that both terms are roughly equally balanced

N_{cls} = Number of anchors in minibatch (~ 256)
N_{reg} = Number of anchor locations (~ 2400)

How to Train?

```
1. train_rpn(M0) → M1                    # Train an RPN initialized from M0, get M1

2. generate_proposals(M1) → P1           # Generate training proposals P1 using RPN M1

3. train_fast_rcnn(M0, P1) → M2          # Train Fast R-CNN M2 on P1 initialized from M0

4. train_rpn_frozen_conv(M2) → M3        # Train RPN M3 from M2 without changing conv layers

5. generate_proposals(M3) → P2

6. train_fast_rcnn_frozen_conv(M3, P2) → M4   # Conv layers are shared with RPN M3

7. return add_rpn_layers(M4, M3.RPN)          # Add M3's RPN layers to Fast R-CNN M4
```

1. 프리트레인 된 M0을 (VGG) -> M1으로 바뀜
2. M1을 통해 ROI로 뽑음 그게 P1
3. M0랑 P1만 사용해서 Fast RCNN으로 학습 시킴
4. M2를 가져와서 다시 RPN을 학습 시킴 그 결과가 M3
5. M3에서 Proposal을 뽑아냄. 그게 P2
6. M3와 P2를 이용해서 Faster RCNN을 돌림. 최종 결과 M4

Result

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	**0.2 seconds**
(Speedup)	1x	25x	**250x**
mAP (VOC 2007)	66.0	**66.9**	**69.9**

빨라 짐 + 성능 개선

Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**

Result

Table 1: the learned average proposal size for each anchor using the ZF net (numbers for $s = 600$).

anchor	128^2, 2:1	128^2, 1:1	128^2, 1:2	256^2, 2:1	256^2, 1:1	256^2, 1:2	512^2, 2:1	512^2, 1:1	512^2, 1:2
proposal	188×111	113×114	70×92	416×229	261×284	174×332	768×437	499×501	355×715

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

Result

Table 2: Detection results on **PASCAL VOC 2007 test set** (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF, but using various proposal methods for training and testing.

train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2000	SS	2000	58.7
EB	2000	EB	2000	58.6
RPN+ZF, shared	2000	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2000	RPN+ZF, unshared	300	58.7
SS	2000	RPN+ZF	100	55.1
SS	2000	RPN+ZF	300	56.8
SS	2000	RPN+ZF	1000	56.3
SS	2000	RPN+ZF (no NMS)	6000	55.2
SS	2000	RPN+ZF (no *cls*)	100	44.6
SS	2000	RPN+ZF (no *cls*)	300	51.4
SS	2000	RPN+ZF (no *cls*)	1000	55.8
SS	2000	RPN+ZF (no *reg*)	300	52.1
SS	2000	RPN+ZF (no *reg*)	1000	51.3
SS	2000	RPN+VGG	300	59.2

Result

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**

Q & A

들어주셔서 감사합니다.