Faster R-CNN 논문 리뷰

: Towards Real-Time Object Detection with Region Proposal Networks

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
기존 Fast R-CNN의 한계점

1. CPU 기반의 selective search로 인해 detection network에서 **region proposal**에 사용되는 **시간이 많이 소요**됨.

2. Region proposal을 수행하는 **Selective search**가 **외부 모듈**로 존재함

Real-time을 요구하는 task를 충족시킬 수 없었음

완전한 **end-to-end** 모델이 아님

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Faster R-CNN에서의 해결방안

CPU 기반의 **Selective Search**

느린 수행 속도

➡️ cpu기반 - 이미지당 2sec 소요

외부모듈

➡️ deep-neural network x
Fast-RCNN의 외부 모듈로 존재

GPU 기반의 **'Region Proposal Network'**

proposal 연산

➡️ 거의 cost-free 수준의 비용 감소

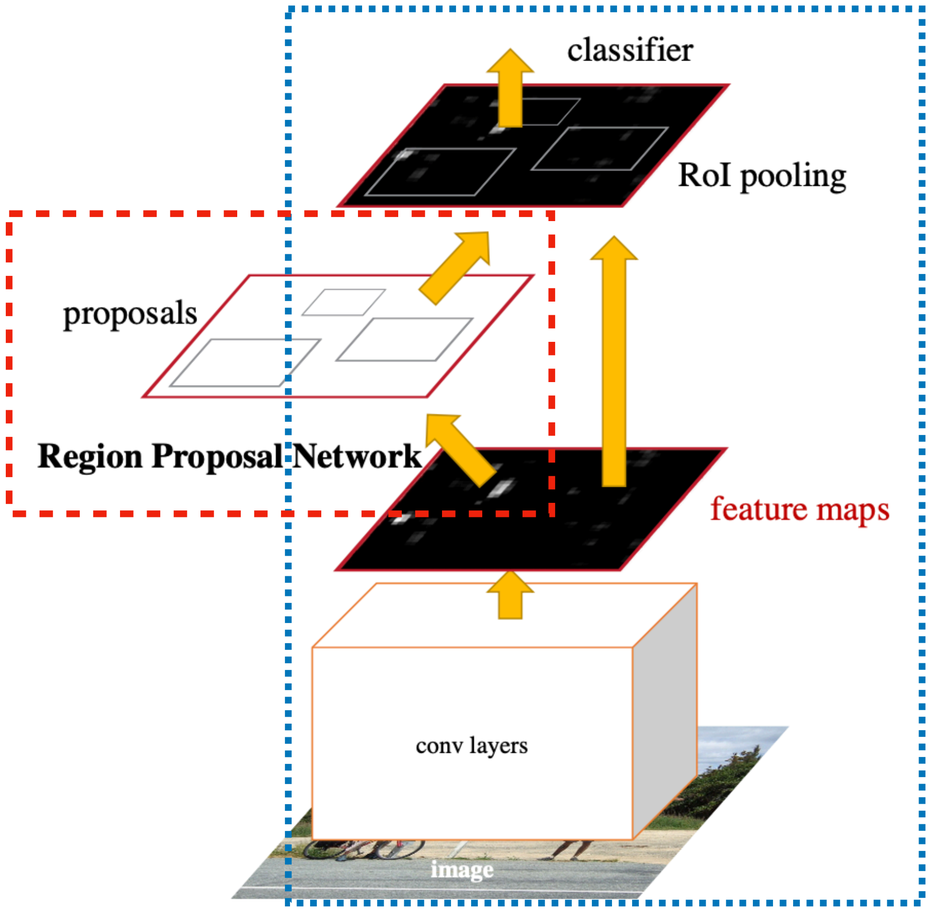
빠른 수행 속도

➡️ gpu기반 - 이미지당 10ms 소요

end-to-end모델 실현

➡️ Deep-Learning Network ✅
convolutional features를 공유

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



Faster R-CNN은 two modules로 구성됨

1. Selective search ➡ RPN

   Region proposal(후보 추천)

   Fully convolutional networks

2. Fast R-CNN detector

   추천받은 영역에 대해 classification + bb regression

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
RPN(Region Proposal Network)

Input = image(of any size)

$$t_i = x, y, w, h$$

$$p_i = 0.87\ldots$$

Output =
set of rectangular object proposals
+ objectness scores

Fully Conv network로 구성하여 모델링함

Why?

Fast R-CNN과 연산을 **공유**하는 것이 궁극적 목표이기 때문

So, 연구진들은 2개의 네트워크가 공통된
Conv layers 세트를 공유한다고 가정함

ZF(5개의 공유가능한 conv layer로 구성)와
VGG-16(13개의 공유가능한 conv layer로 구성) 모델을 참고

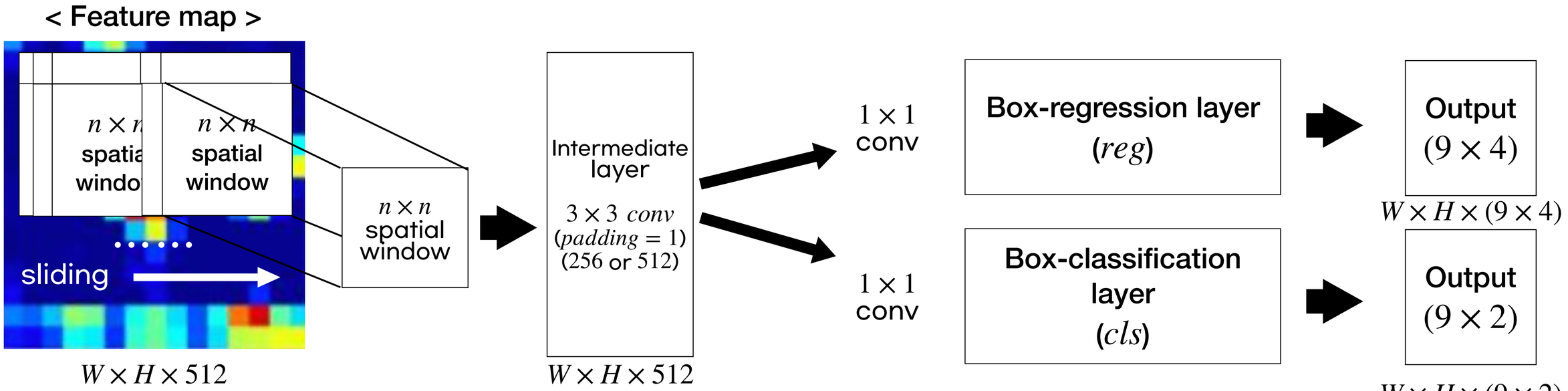
Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
RPN(Region Proposal Network)

(shared 된 마지막 Conv layer 를 통과한)

Region proposal을 하기 위해 작은 네트워크를 feature map 위로 슬라이딩 시킴

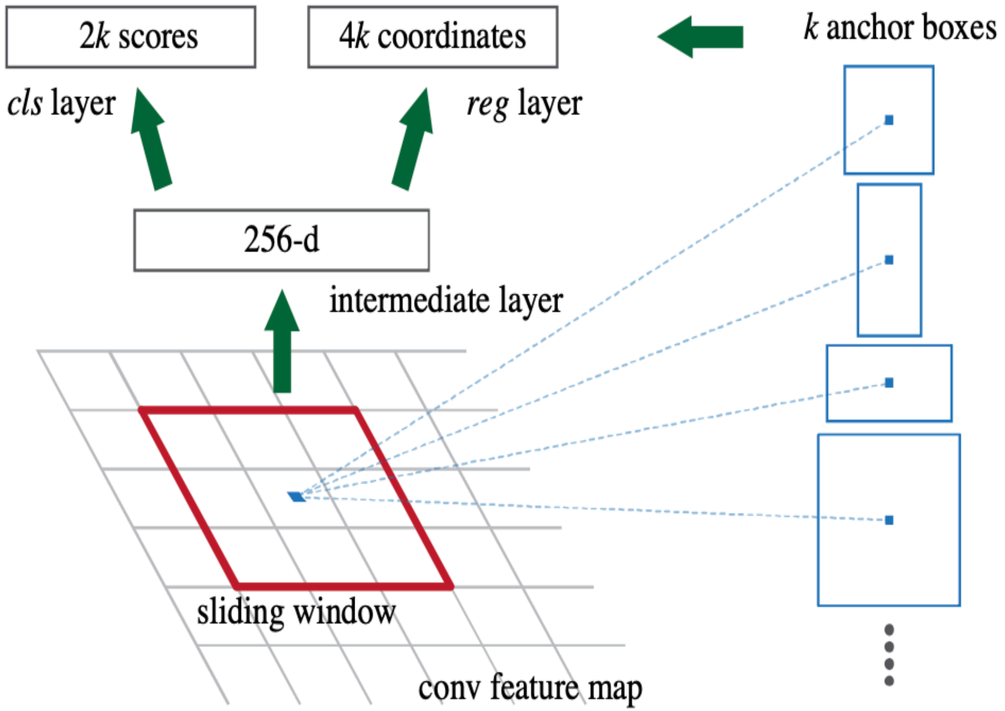
feature map의 $n \times n$ 의 spatial window를 input으로 받아 계산된 결과를 각각

2개의 동일한 fully connected layer인 box-regression layer & classification layer에 전달



$n = 3$ (본 논문에서는 입력 이미지의 effective recptive field가 크다는 점에 주목하여 n=3을 사용)

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
RPN(Region Proposal Network)



3×3 filter를 통한 convolution 과정(sliding windows)에서

각각의 위치에서 가능한 최대 region proposal 수 = k로 표시,
k proposal은 k reference box에 대한 매개변수 = 'Anchor"

Sliding 되는 각 window위치에
미리 defined 된 k개의 Anchor boxes가 적용되어

2개의 동일한 1×1 $conv$ (cls, reg)를 거쳐
각각의 Anchor box마다 2개의 output(object일 확률 & 좌표값) 가짐

즉, 결과적으로 각 sliding window 마다
$2k$개의 objectness score와 $4k$개의 box coordinate가 산출됨.

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Anchor box

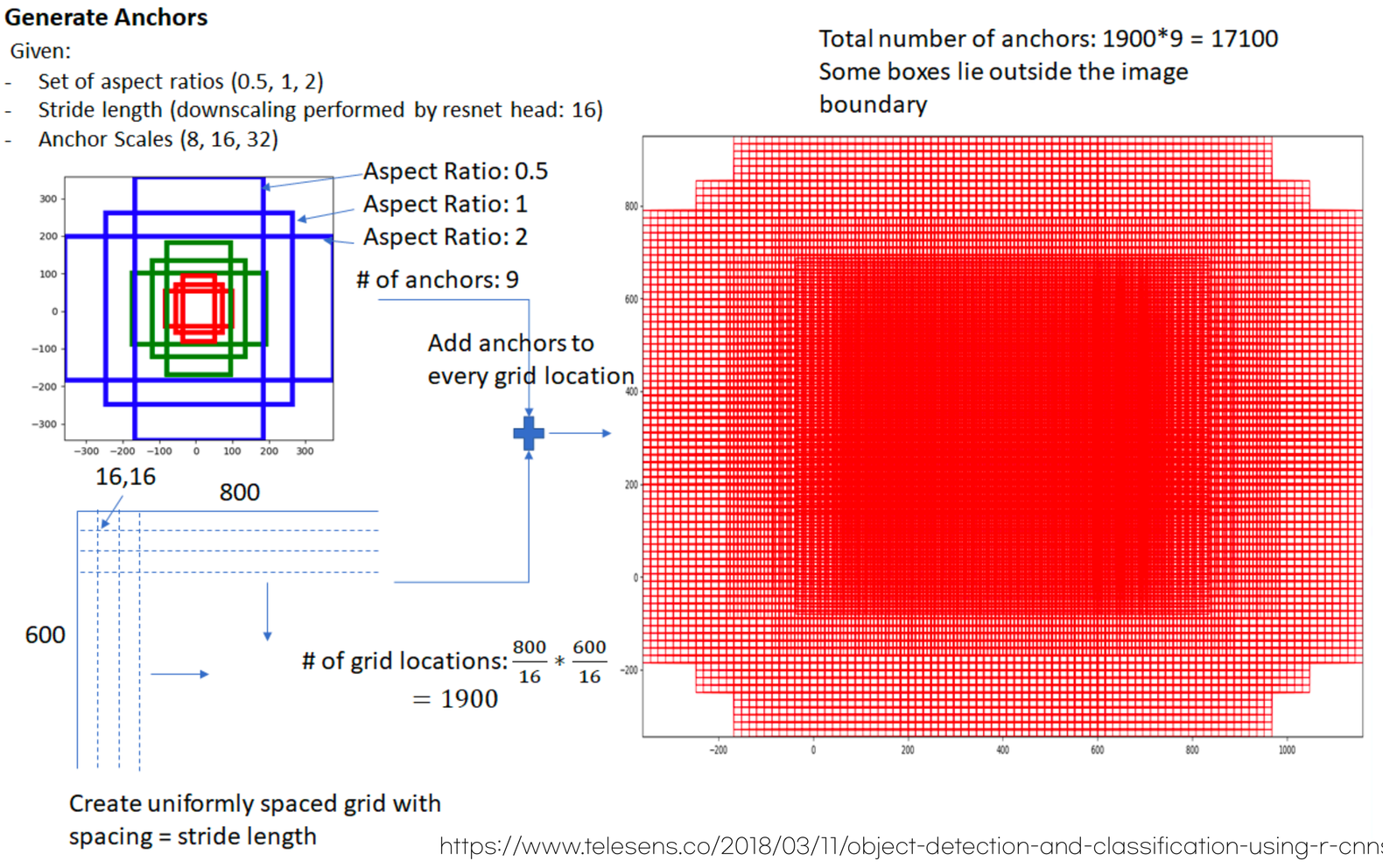
Anchor box

= sliding window의 각 위치의 BB의 후보상자

= Anchor는 window의 중앙에 위치, 여러 scale & aspect ratios를 가짐

3 scale		3 aspect ratios
128×128		$1:1$
256×256	✕	$1:2$
512×512		$2:1$

= 9 anchors

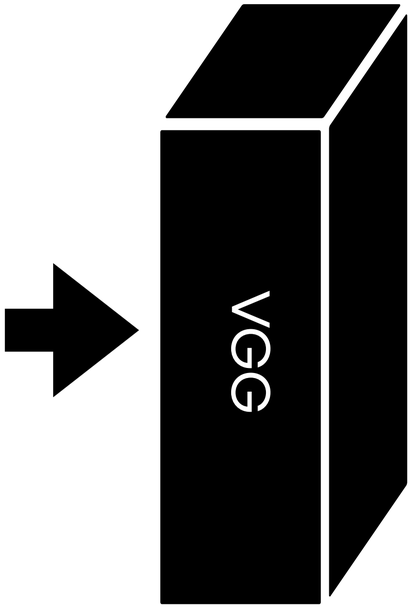


https://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Anchor box

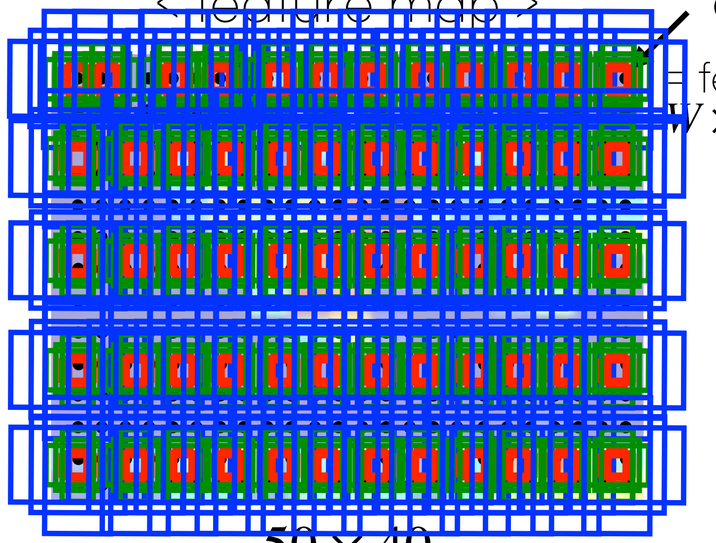
< image >

800×600



subsampling ratio
= $\frac{1}{16}$ (vgg)

< feature map >



Anchor center

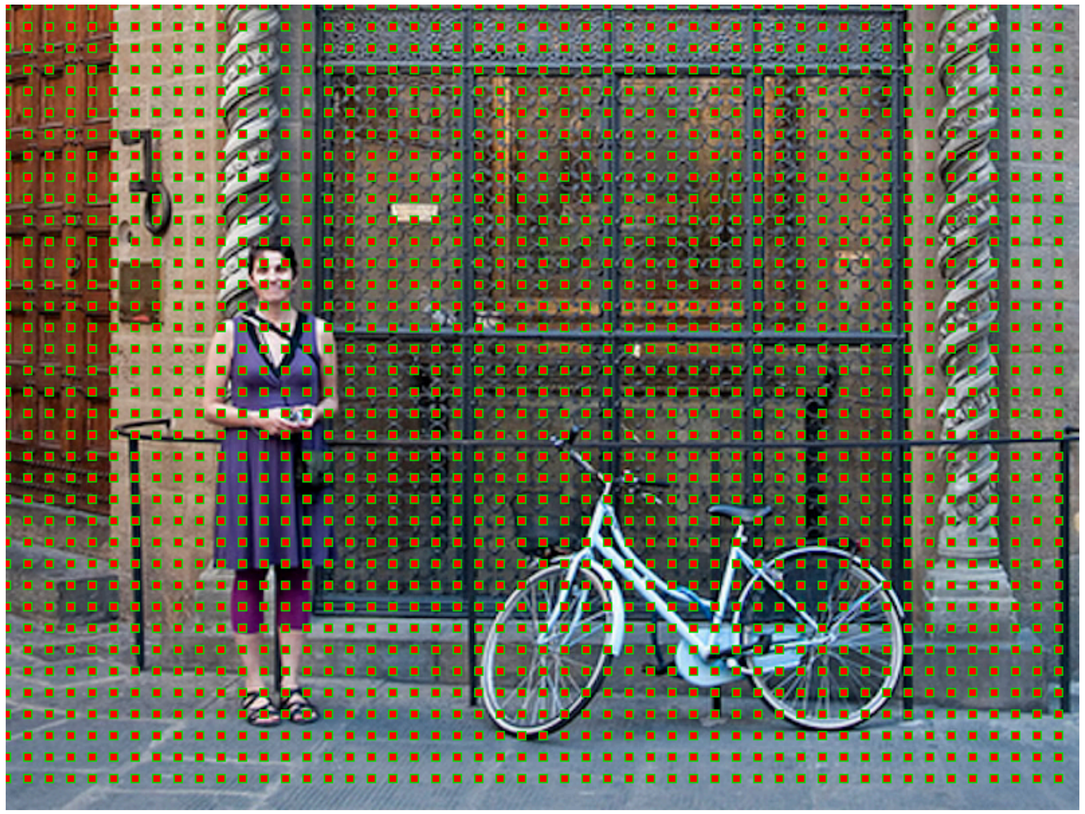
= featrue map의 $W \times H$만큼 존재

50×40

피쳐맵의 개별 그리드마다 k개의 anchor box가 생성됨

만약 feature map의 size가 $W \times H$라면
Anchor box의 개수는 $W \times H \times k$가 됨

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Anchor box



원본이미지에 대입된 Anchor center

중간 : 개별 point에 대한 anchor 영역

왼쪽 : anchor box 영역

오른쪽 : 모든 anchor 영역

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Translation-Invariant

RPN의 중요한 속성은 앵커와 앵커에 상대적인 proposal을 계산하는 함수 측면에서
모두 Translation-Invariant 하다는 점이다.

만약 이미지의 object를 translate(이동되었을 때)하면, proposal은 translate되어야 하고,
동일한 함수는 어느 위치에서든 제안을 예측할 수 있어야 한다.

이러한 Translation-Invariant한 속성이 본 논문의 방식(Anchor)에서는 보장된다.

이와 비교하여 Multibox방법은 Translation-Invariant하지 않은 800개의 anchors들을 만들어 내는데
K-means 를 사용한다.

그래서 Multibox는 object가 translate(이동되었을 때) 동일한 proposal을 생성하는 것을 보장하지 못한다.

또 이러한 속성은 모델의 사이즈를 줄여줄 수 있다.

RPN : 2.8×10^4 parameters, $512 \times (4+2) \times 9$

Multibox : 6.1×10^6 parameters, $1536 \times (4+1) \times 800$

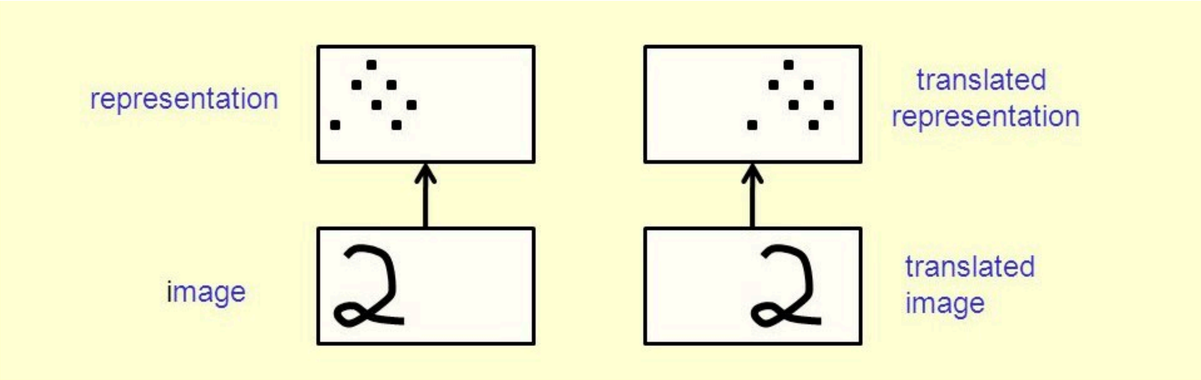
Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Translation-Invariant

translation equivariance

- Equivariance란, 함수의 입력이 바뀌면 출력 또한 바뀐다는 뜻이고, translation equivariance는 입력의 위치가 변하면 출력도 동일하게 위치가 변한채로 나온다는 뜻이다.
- Convolution 연산은 translation equivariance 특성을 갖고 있다.

translation invariance란 input의 위치가 달라져도 output이 동일한 값을 갖는것을 말한다

- Equivariance와 invariance는 서로 반대되는 개념
- translation invariance는 입력의 위치가 변해도 출력은 변하지 않는다는 의미로, 강아지 사진에 강아지가 어느 위치에 있건 상관없이 그냥 강아지라는 출력을 하게 된다는 의미다.

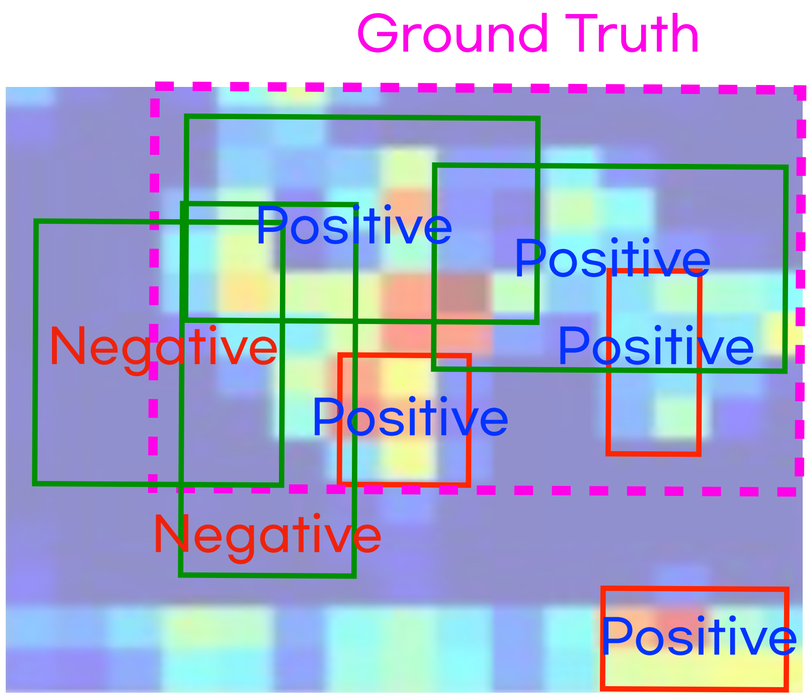


논문에서는 Anchor가 translation invariance한 속성을 가지고 있기 때문에, object가 이동해도 동일한 proposal을 생성한다고 함.

= 균일한 간격, 일정한 규칙으로 생성되어, 물체가 특정 위치에 존재할 때만 탐지가 잘 되거나, 혹은 특정 위치에서는 탐지가 잘 되지 않는 것이 아닌 모든 위치에서 동일하게 proposal이 가능하다.

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Anchor targeting - Positive anchor & Negative anchor

anchor_target_layer에 해당되며 RPN train할때만 적용되며, 실제 classification에는 사용되지 않음



하나의 ground truth box가 여러개의 anchor에 대해 positive label을 부여할 수 있음

RPN(classifier, regressor)을 학습시키기 위해 Ground Truth와 비교하여 각 anchor마다 binary class labeling 함

Positive		Negative
∘ 가장 높은 IoU값의 anchor ∘ 0.7보다 높은 IoU값의 anchor	&	∘ 0.3보다 낮은 IoU값의 anchor

Positive와 Negative 둘 다 해당하지 않는 Anchor boxes들은 학습에서 제외시킴.

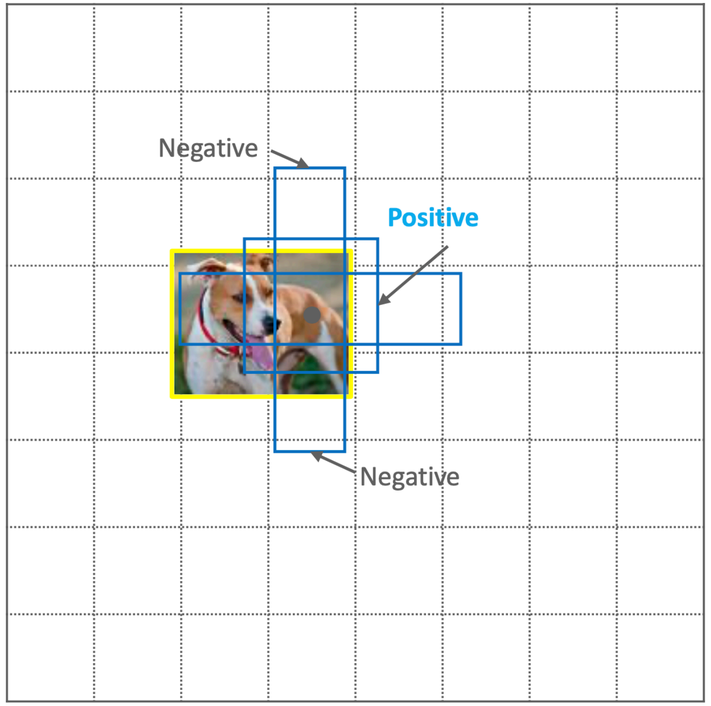
→ 라벨링된 anchor boxes들은 이후 학습을 위한 training data로 쓰임

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Prediction - Classification & Bounding box Regression

Classification = *cls layer*

Output = anchor당 2개의 예측값
FG or BG softmax 확률값

Positive, negative를 예측해서 classification loss를 줄여가는 방향으로 학습



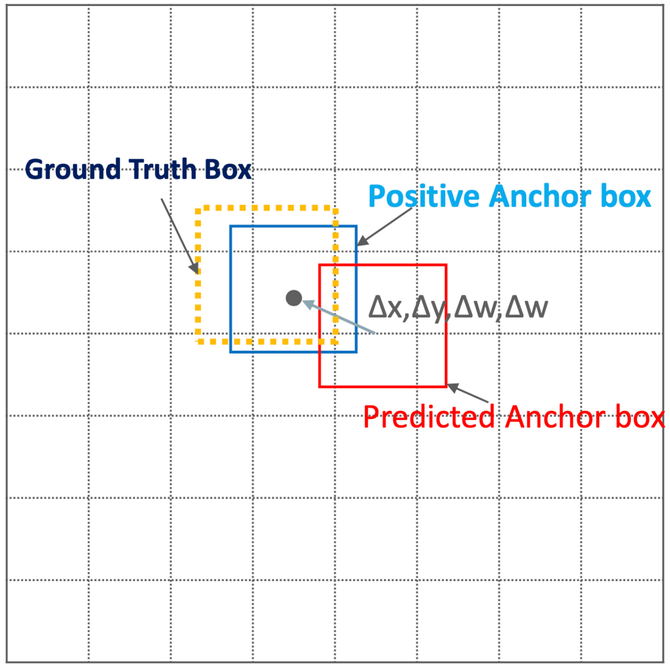
Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Prediction - Classification & Bounding box Regression

Bounding box Regression = $reg\ layer$

Output = anchor의 $\Delta_x, \Delta_y, \Delta_w, \Delta_h$ 총 4개의 값

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

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



t_x, t_y : 박스의 center좌표

t_w, t_h : 박스의 너비,높이

x, y, w, h : predicted box

x_a, y_a, w_a, h_a : anchor box

x^*, y^*, w^*, h^* : ground-truth box

실제로 region proposal할 predicted box를
Positive anchor box(IoU가 높은)와의 차이를 최소화하는
방식으로 BB regression을 수행

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
RPN loss function

$$smooth_{L1} = \begin{cases} 0.5x^2 & if\,|x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

predicted value

$$L(\{p_i\},\{t_i\}) = \underbrace{\frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*)}_{\text{Classification loss}} + \underbrace{\lambda\frac{1}{N_{box}}\sum_i p_i^* \cdot L_{reg}(t_i, t_i^*)}_{\text{Bounding box Regression loss}}$$

Ground – truth

L_{cls} = 두 클래스간 log loss
cross_entropy(predicted _class, actual_class)

$L_{reg} = smooth_{L1}$ loss 함수
$p_i^* = 1$일 때만(positive일 경우) 해당

i = Anchor의 index

p_i = Anchor_i의 softmax 예측값(foreground or background)

p_i^* = Anchor_i의 Ground truth의 lable값

$$p_i^* = \begin{cases} 1 & \text{anchor positive = foregorund} \\ 0 & \text{anchor positive = backgorund} \end{cases}$$

N_{cls} = normalization. mini-batch 크기와 동일

RPN Loss는 후에 Fast R-CNN의 Loss와 함께 학습됨.

t_i = Anchor_i의 예측 bb 좌표(벡터)

t_i^* = Anchor_i의 실제 bb 좌표(벡터)

N_{reg} = normalization. anchor locations의 갯수

λ = 가중치 hyperparameter, 10

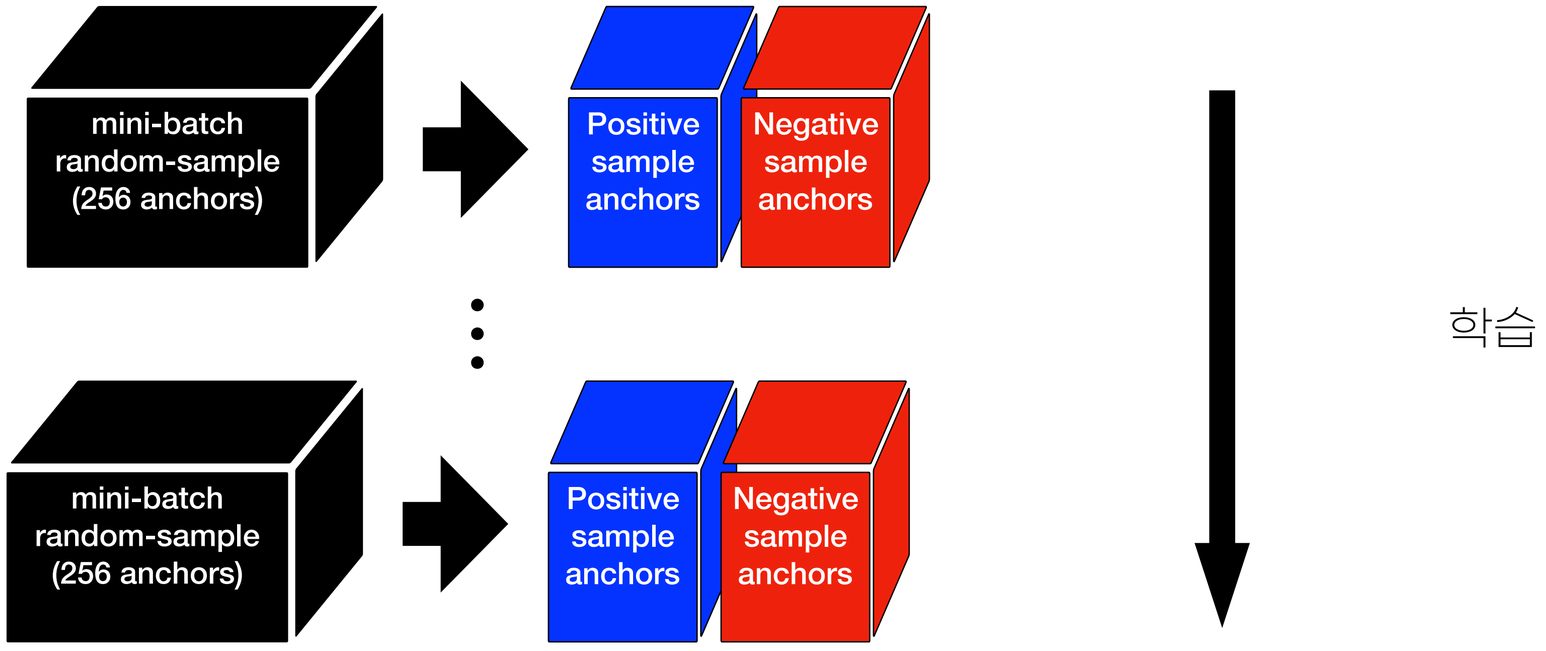
람다값이 매우 큰 영향을 미치지는 않기 때문에 적절한 값을 설정

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Training RPNs

RPN은 end-to-end 방식으로 back propagation & SGD에 의해 학습이 진행

Random하게 학습시킬시 negative sample이 더 많을 경우 편향된 학습이 되기 때문에

mini-batch로 학습시 positive sample, negative sample을 1:1 비율로 맞춰줌



현실적으로 1:1 비율을 지속적으로 유지시키는 것은 매우 어렵

이 경우 zero-padding을 시켜주거나, 아예 없는 경우는 IoU값이 가장 높은 값을 사용

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Objectness Score, NMS

RPN proposal이 겹치는 경우(overlap)가 많아
중복성을 줄이기 위해 NMS를 적용

Top-N ranked proposal regions를 사용

이미지당 2,000개의 proposal

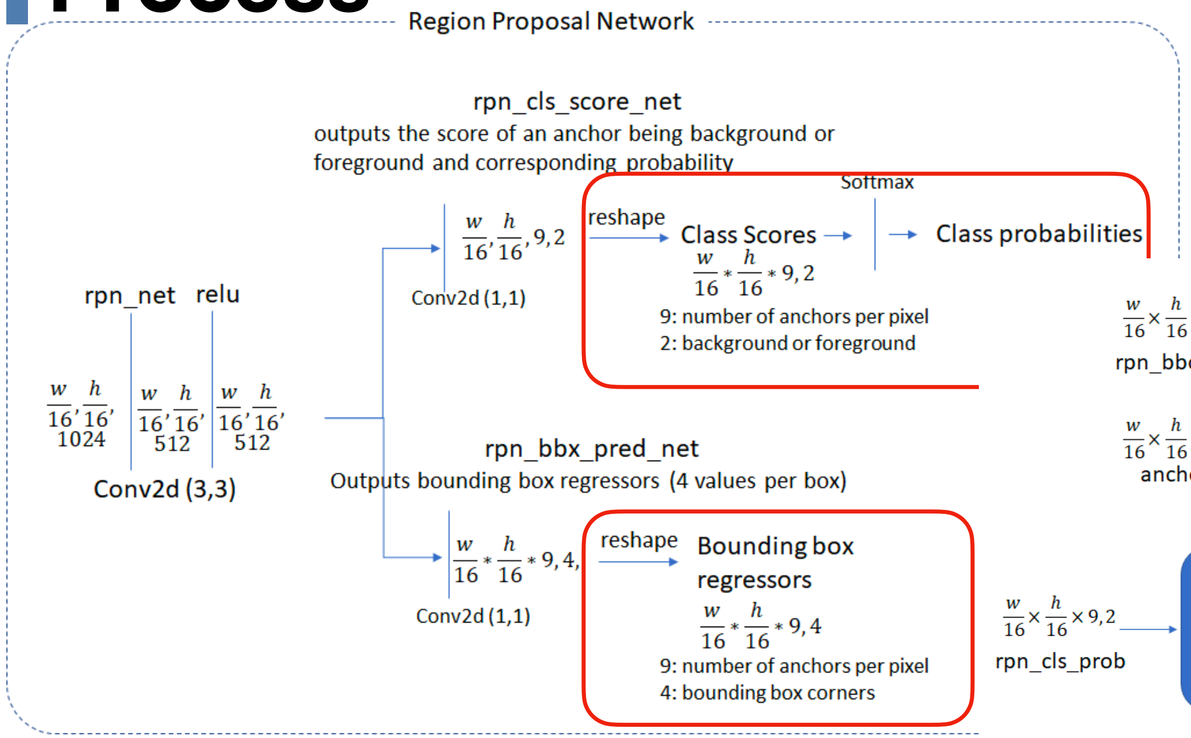
train시에는 2,000개, evaluate시에는 300개
Top-N의 개수는 유연하게 사용 가능함

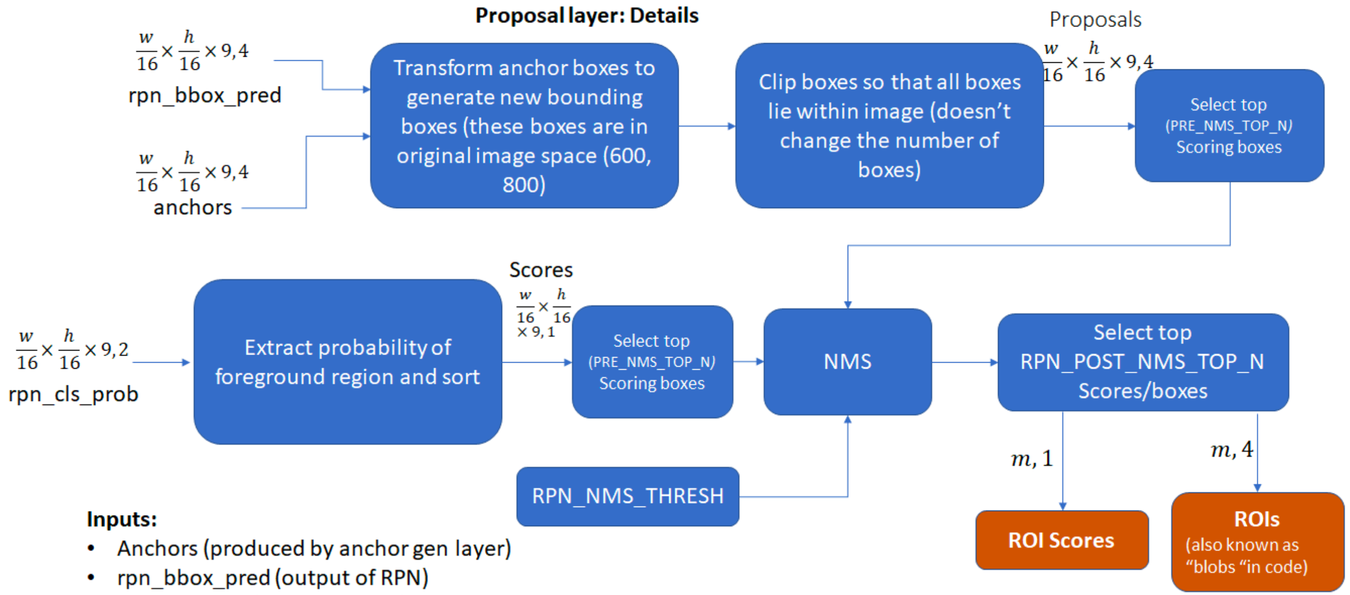
Objectness score : predicted box가 **object일 확률**(softmax) × **IoU값**(with ground-truth bbox)

Objectness score가 높은 순으로 Region proposal box 추출

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

Process





Inputs:

- Anchors (produced by anchor gen layer)
- rpn_bbox_pred (output of RPN)
- rpn_cls_prob (output of RPN)

Outputs:

- ROIs (pruned anchors after applying bbox regression coefficients)
- ROI scores

Parameters:

- RPN_NMS_THRESH: Threshold used in non-maximum suppression
- RPN_PRE_NMS_TOP_N: Number of scores/ROI boxes to send to NMS
- RPN_POST_NMS_TOP_N: Number of boxes to retain after NMS

$m = \min($number of output boxes from NMS, RPN_POST_NMS_TOP_N$)$

- Depending on the NMS threshold (RPN_NMS_THRESH), number of output boxes after NMS may be < RPN_POST_NMS_TOP_N

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Sharing Features for RPN and Fast R-CNN

독립적으로 train된 RPN & Fast R-CNN은 Conv layer를 각각 다른방식으로 수정하게 됨.

그렇다면 이 두가지의 Network를 어떻게 train할 것인가

그래서, 두 네트워크 간 Conv layer를 sharing하는 기술을 개발해야 했는데,
본 논문에서 고려한 방법은 다음 3가지와 같음.

<Alternating training>

- Train RPN
- Proposal -> Fast RCNN
- Initialize RPN
- Iterated process

<Approximate joint training>

- Merge into one network (RPN & Fast RCNN)
- RPN loss + Fas tRCNN loss
- Easy to implement
- Ignore derivative, proposal boxes' coordinates
- Approximate result
- Reduce training time 25-50% than Alternating training

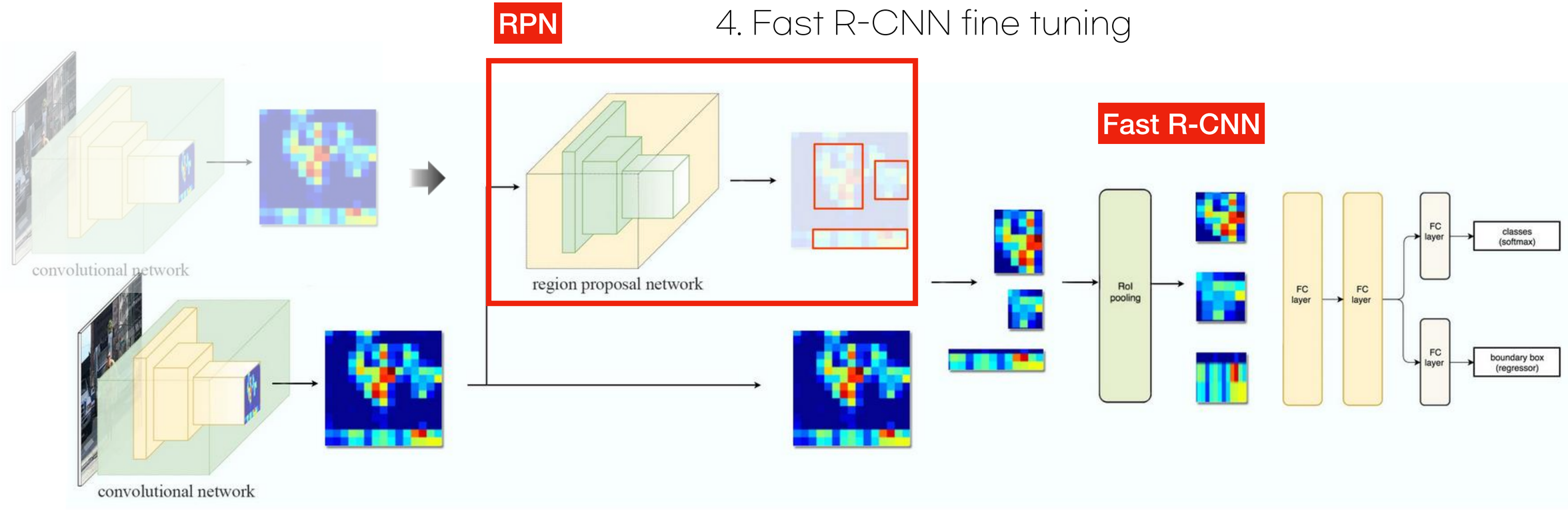
<Non-Approximate joint training>

- Enable derivative, proposal boxes' coordinates RPN
- nontrivial problem

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
4-step Alternating Training

Alternating Training

1. RPN을 먼저 학습
2. Fast R-CNN classification, Regression학습
3. RPN fine-tuning
4. Fast R-CNN fine tuning



Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - SS, EB, RPN(ZF & VGG)

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

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - RPN + VGG(1)

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**

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - RPN + VGG(2)

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**

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - Timing

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**

ZF > VGG > SS순으로 처리속도가 빨랐으며, VGG에 비해 ZF 모델이
더 가볍기 때문에 수행속도에 있어서 더 우수하지만, 정확도 면에서는 VGG가 앞섬

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - Anchor box

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**

다양한 scale, 다양한 aspect ratios를 사용한 Anchor에서 성능이 좋았음

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - Lambda

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

λ 값을 10으로 설정했을 때, 성능이 가장 좋았으며, λ 값은 기본적으로 intensitive 하기 때문에 적절한 값을 설정하면 좋은 결과값을 얻을 수 있음

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - Recall-to-IoU

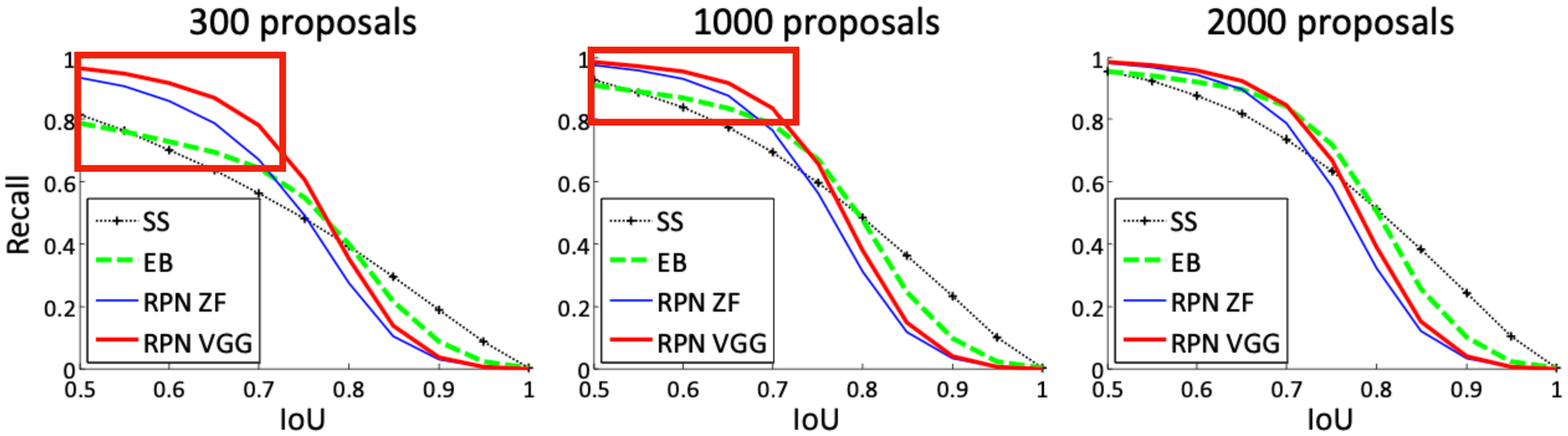


Figure 4: Recall *vs.* IoU overlap ratio on the PASCAL VOC 2007 test set.

Ground-truth의 다른 IoU 비율에 따른 proposal의 재현율
보다 적은 proposal에서도 타 모델에 비해 recall 값이 높음

Faster RCNN : Towards Real-Time Object Detection with Region Proposal Networks
Experiments - One_stage, two_stage

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

One-stage에 비해 Two-stage의 정확도가 더 높음

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

참고자료

Faster RCNN

- Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks
- https://leechamin.tistory.com/221
- https://yamalab.tistory.com/113
- https://rroundtable.github.io/blog/deeplearning/detection/2019/07/26/Faster-RCNN-%EC%A0%95%EB%A6%AC%EA%B8%80.html#translation-invariant-anchors
- http://incredible.ai/deep-learning/2018/03/17/Faster-R-CNN/
- https://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/
- https://github.com/jwyang/faster-rcnn.pytorch/blob/master/lib/model/rpn/anchor_target_layer.py
- https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/
- https://youtu.be/46SjJbUcO-c

참고자료

Translation-Invariance

- https://seongkyun.github.io/study/2019/10/27/cnn_stationarity/
- https://towardsdatascience.com/translational-invariance-vs-translational-equivariance-f9fbc8fca63a
- https://medipixel.github.io/post/anchor-target/?fbclid=IwAR3sCN1gXjcpt0SNcBgCpVsW8Y6jo2u-2MrBkrQGQgy3CSIKkUPPHGt4YY8

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

Thanks