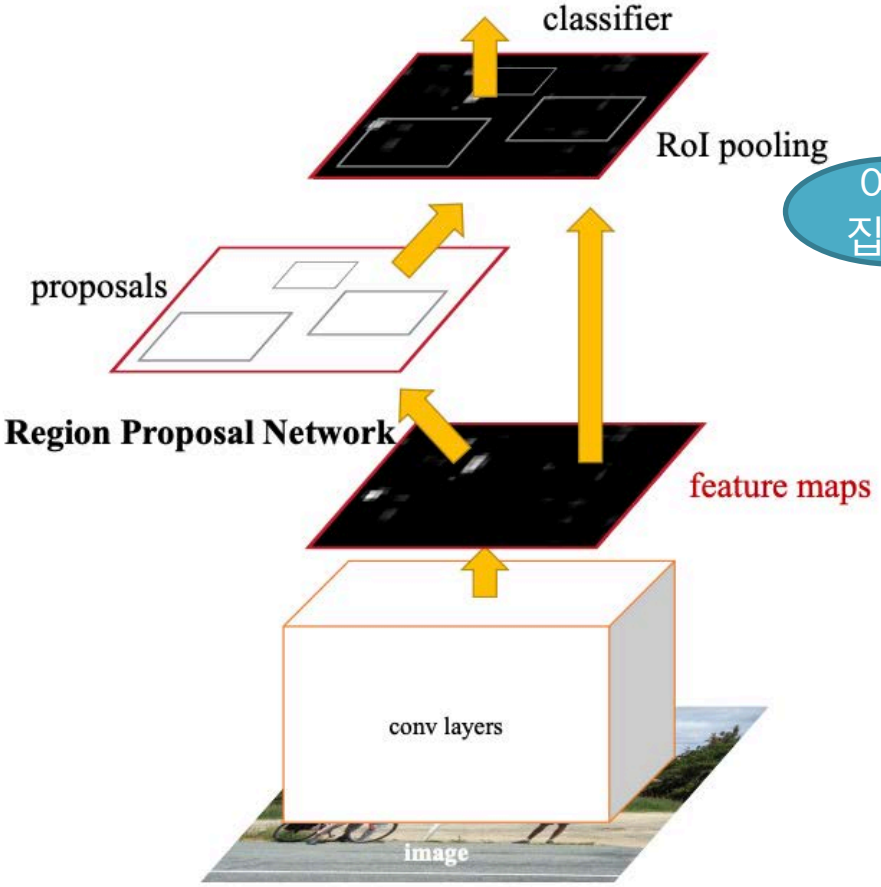
Faster R-CNN

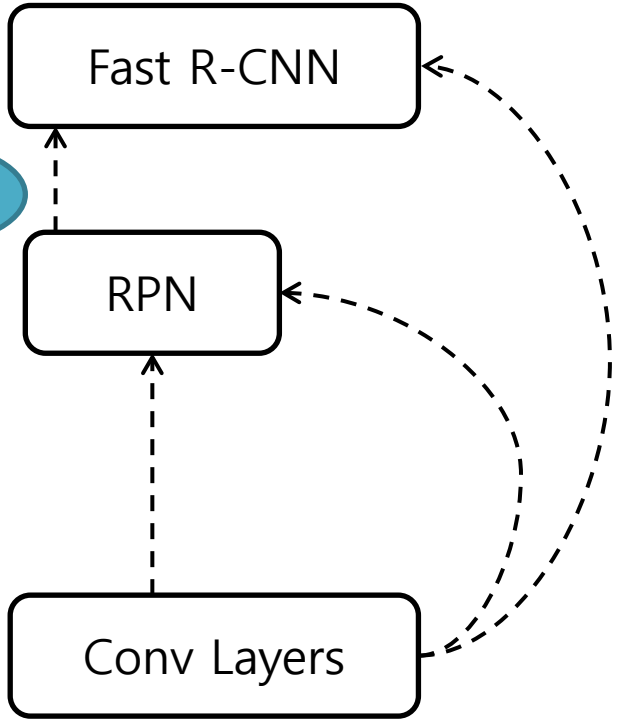
2022.08.10

김정윤

Introduction



여길 집중!!



Conv Features를
공유하도록 network를 쌓자

Introduction

Fast R-CNN

SPP-net

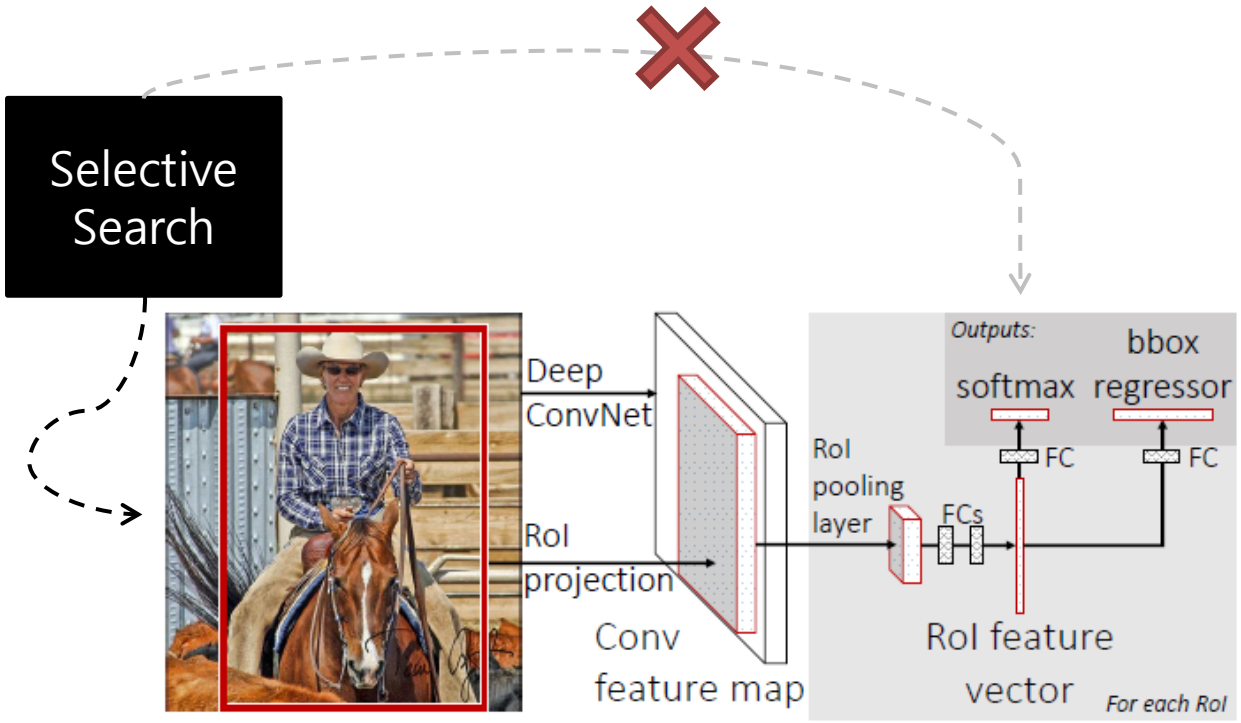
BUT

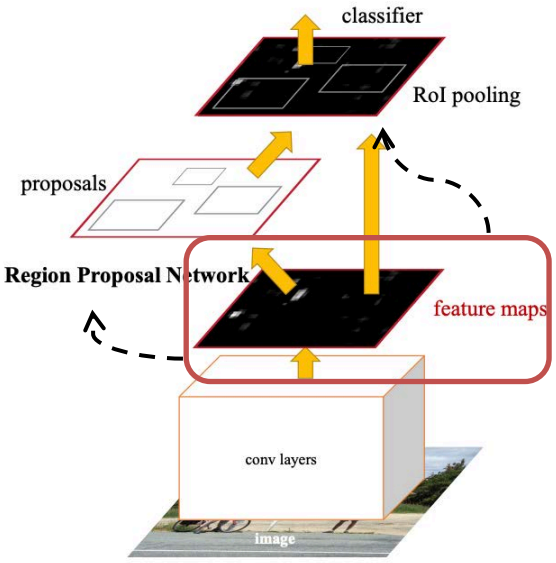
Region proposal이
독립적으로 존재함에 따라
시간이 추가로 걸림
(Bottleneck)

RoI Pooling 방식을 이용해
Proposal을 convolution 간 공유
➔Real-time에 근접한 network

Introduction (idea)

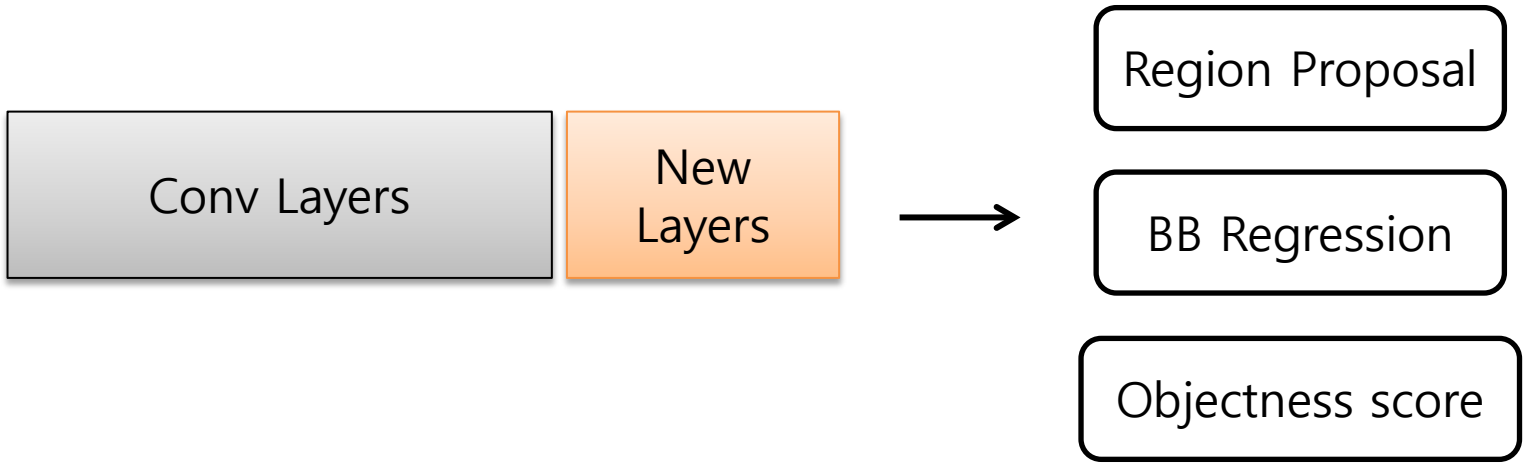
- Region-based detector의 convolution feature map이 Region Proposal에도 쓰일 수 있다



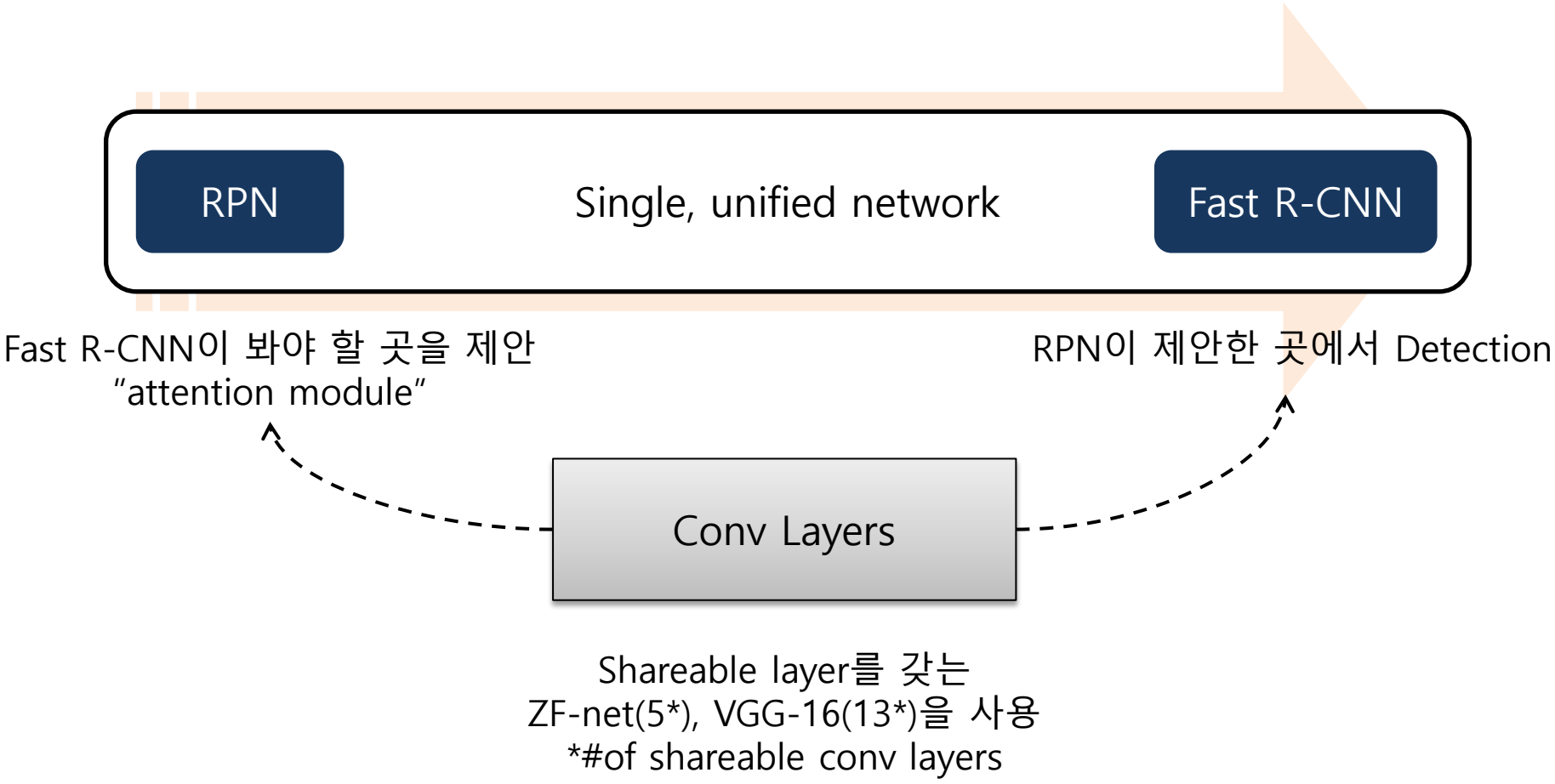


Introduction (idea)

- 기존 conv layer에 추가적인 layer를 쌓아서, region proposal과 regression, objectness score 계산을 동시에 해보자



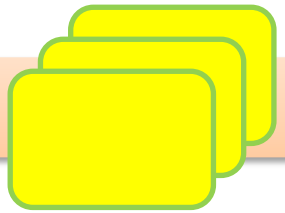
Introduction



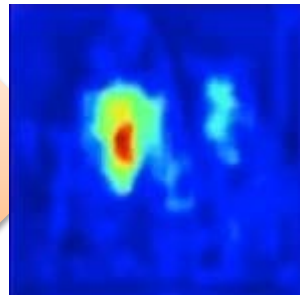
Architecture & Training Process

- Input image를 pre-trained 된 convolutional network에 forward-pass 하여 feature map을 생성

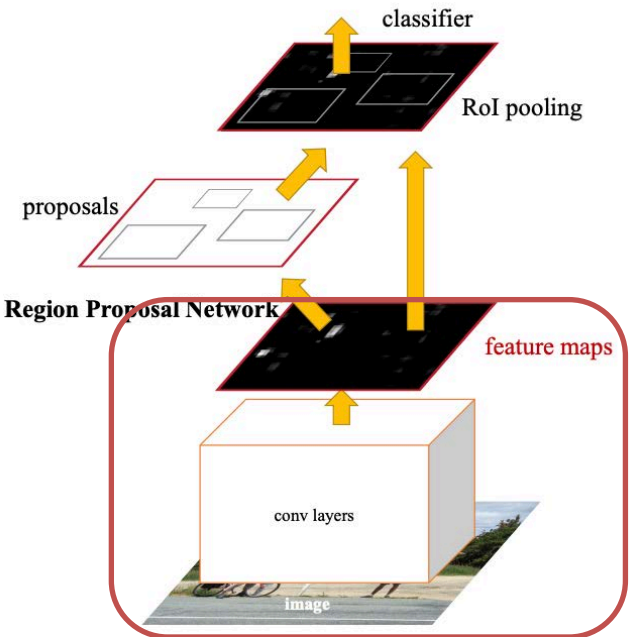
Input Image
600*1000*3
Single scale : h/w 중 짧은 부분이 600 pixel을 갖도록 조정



Conv Layers
VGG-16

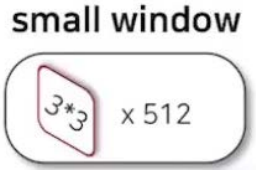


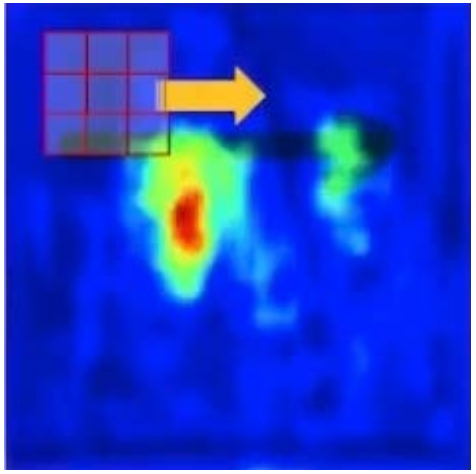
Feature map
40*60*512



Architecture & Training Process

- Feature map에 n*n(논문 : 3*3)의 sliding window를 적용(intermediate layer)시켜 같은 크기의 feature map 생성





Intermediate Layer

Feature map
40*60*512

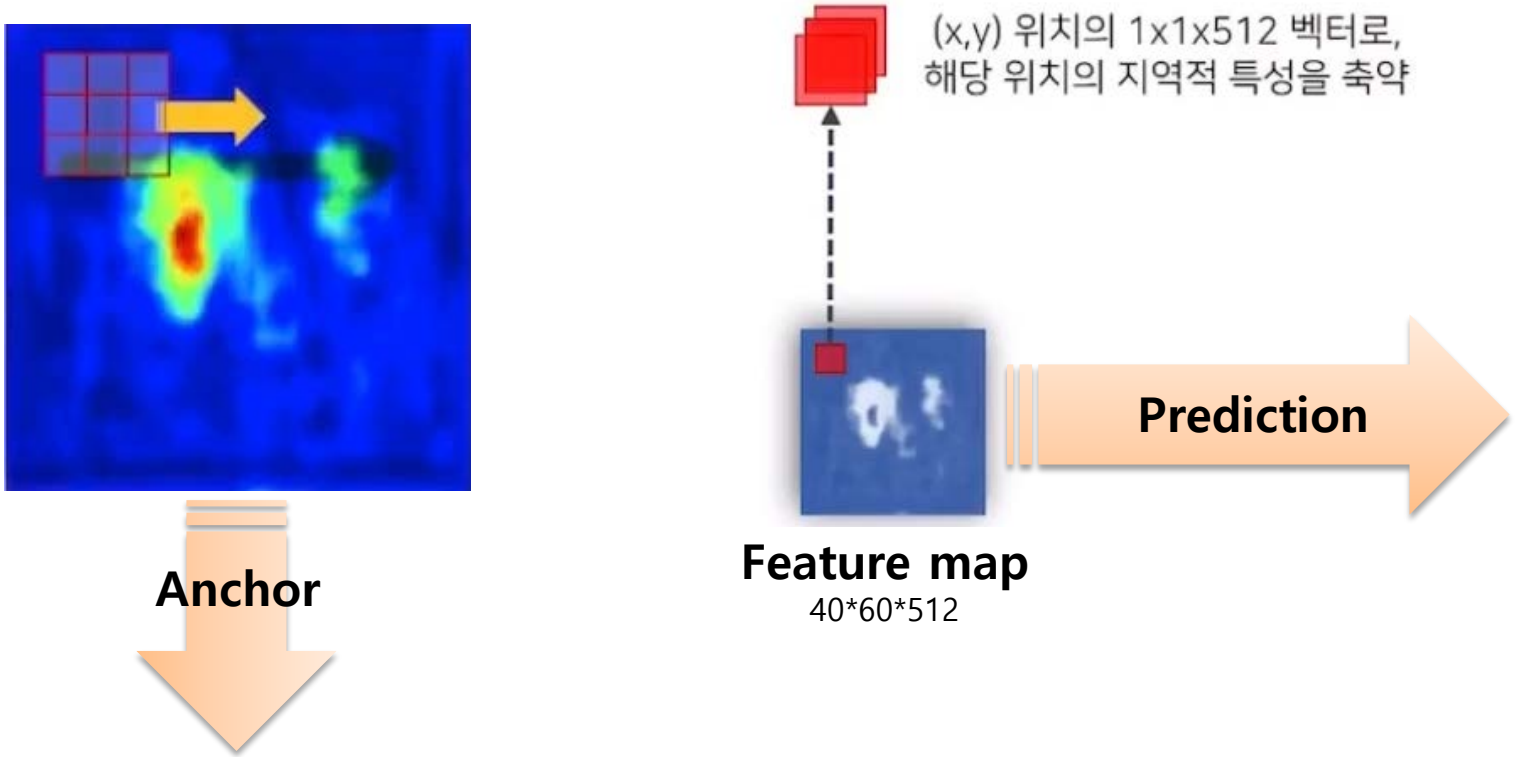
Architecture & Training Process

- 생성된 Feature map의 각 픽셀은 앞선 3*3 sliding window에 mapping됨

Feature map
40*60*512

Architecture & Training Process

- 여기까지가 RPN training의 base단계이며, 이후 anchor를 생성 또는 prediction network를 학습하는 두 갈래로 이해

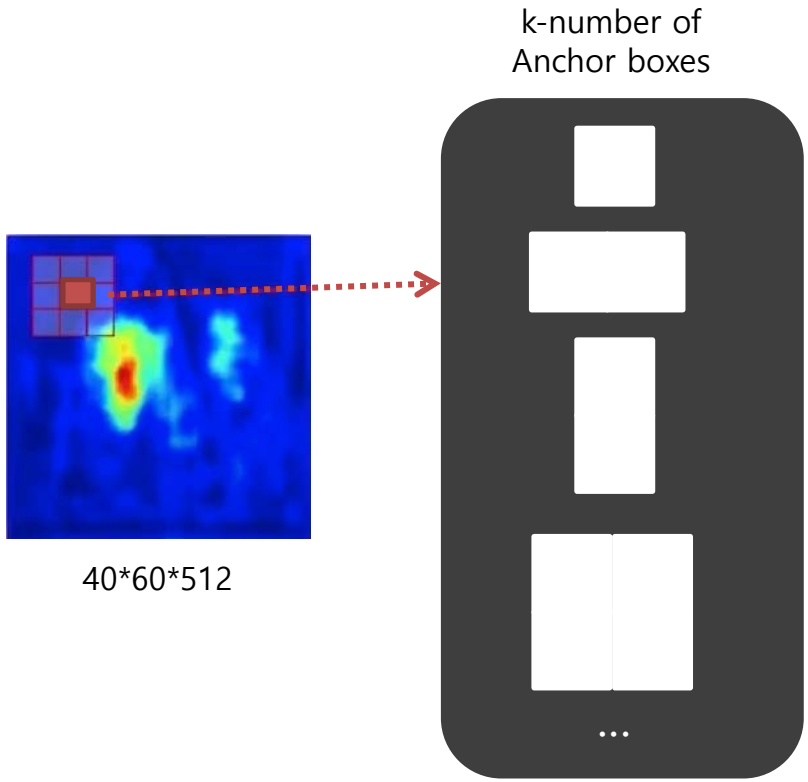


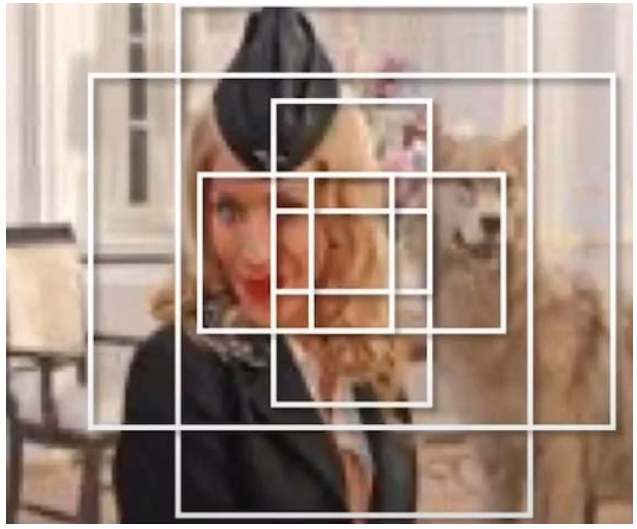
Anchor

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**

- 3*3 sliding window를 적용하는 동시에, 각 window의 중심에 anchor를 배치
- 각 anchor마다 scale*ratio의 k개 anchor box를 생성하며, 이를 input image에 projection

k = scale * ratio





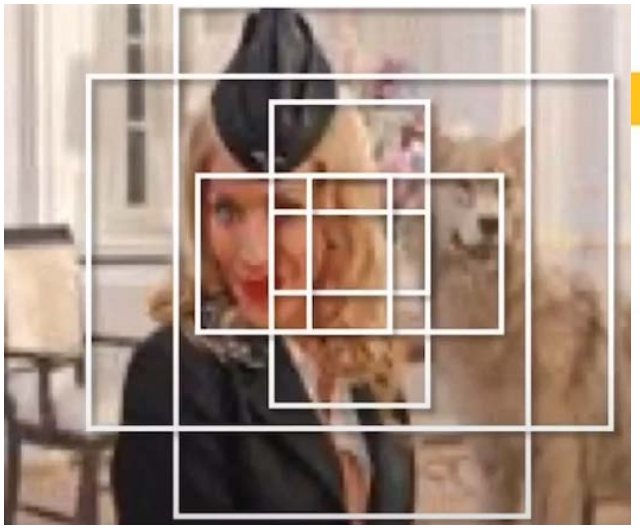
scale	ratio	W	H	Anchor Boxes
3	3	60	40	21600

Anchor

- 생성된 모든 anchor box에 대해 positive, negative, invalid(학습에 쓰이지 않음)로 labeling

$$IoU = \frac{\text{anchor} \cap \text{ground-truth box}}{\text{anchor} \cup \text{ground-truth box}}$$

Objectness Labeling



scale	ratio	W	H	Anchor Boxes
3	3	60	40	21600

21600

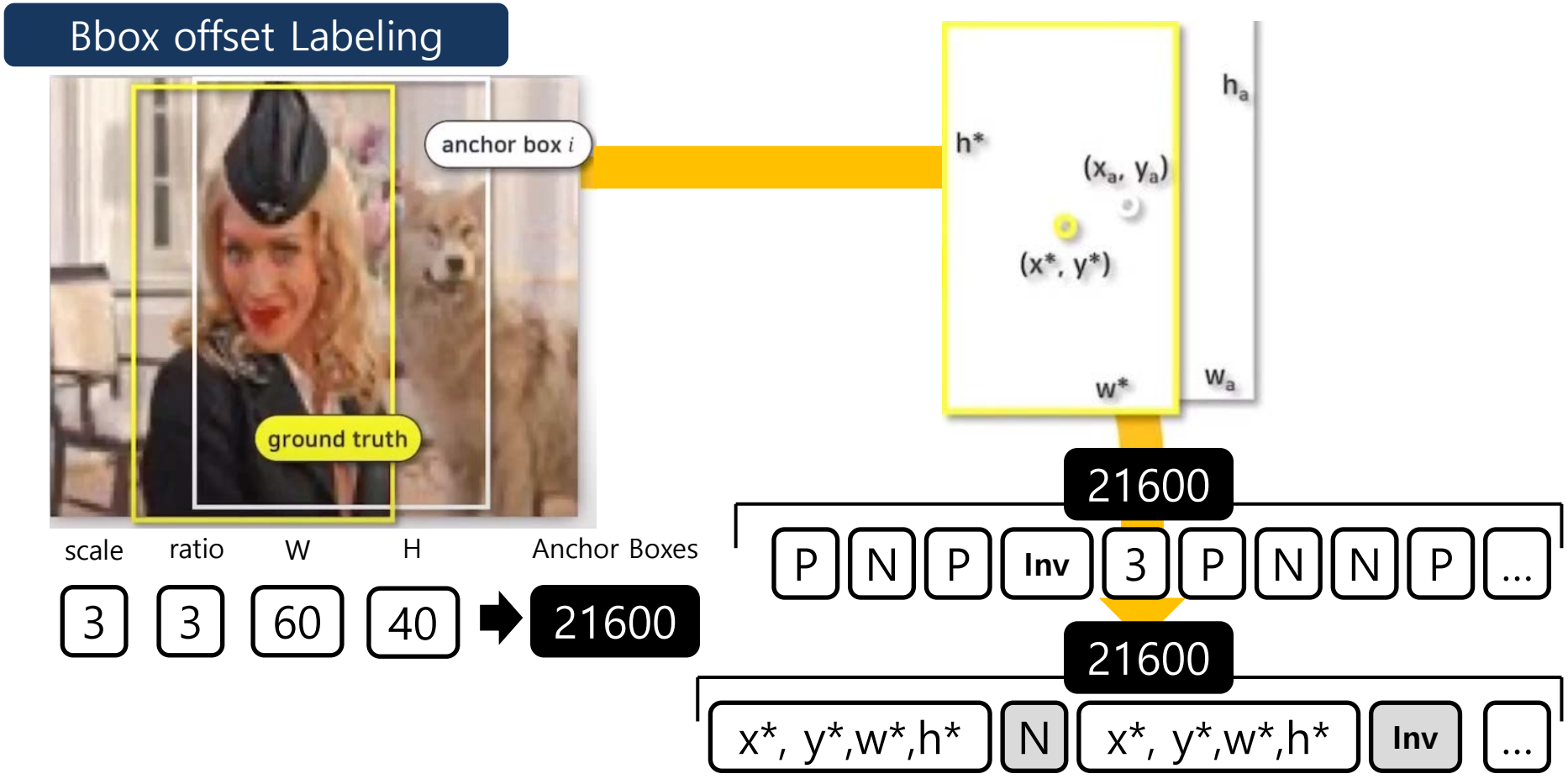
- Positive: 각 Ground truth와 가장 높은 IoU, Ground truth와 IoU 0.7 이상
- Negative: Ground truth와 IoU 0.3 이하
- Invalid: Ground truth와 IoU 0.3 ~ 0.7 사이거나, 이미지 경계를 벗어난 경우

21600

P N P Inv 3 P N N P ...

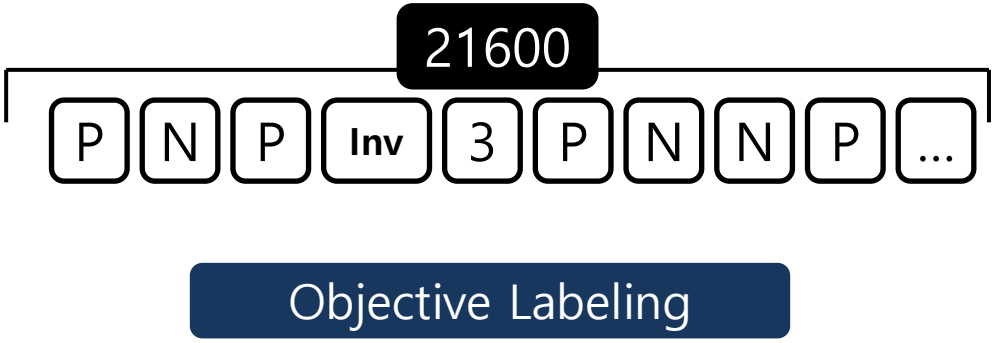
Anchor

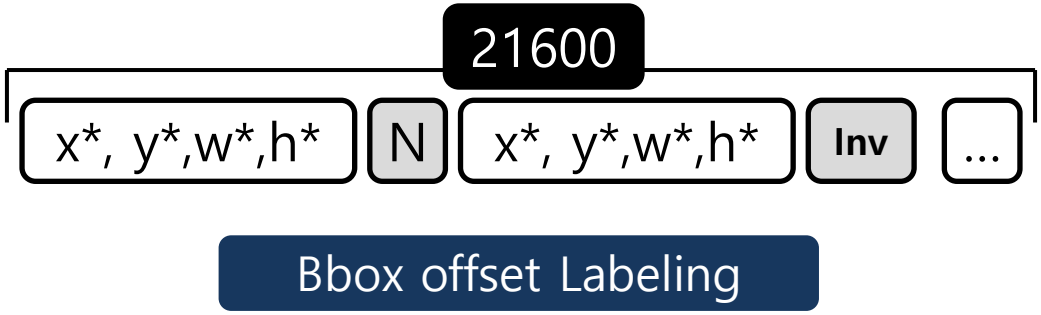
- Positive anchor box에 한해, 해당하는 ground truth의 bounding box offset(x*,y*,w*,h*)를 Labeling



Anchor

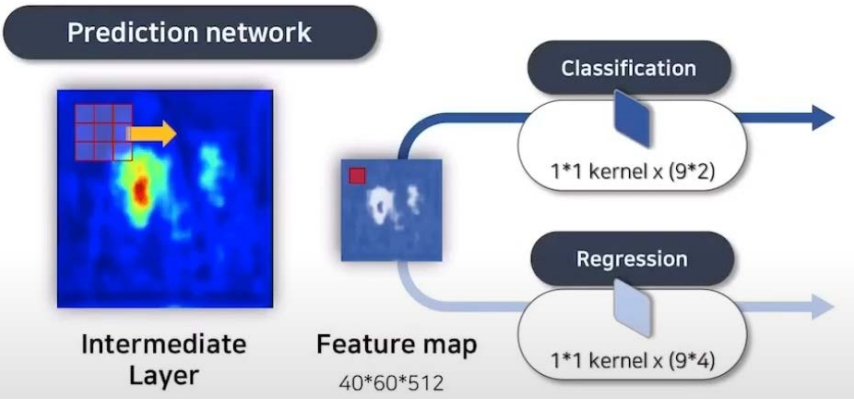
- 결과적으로 3*3 sliding window 단계에서 동시적으로 아래와 같이 anchor box를 생성 및 labeling

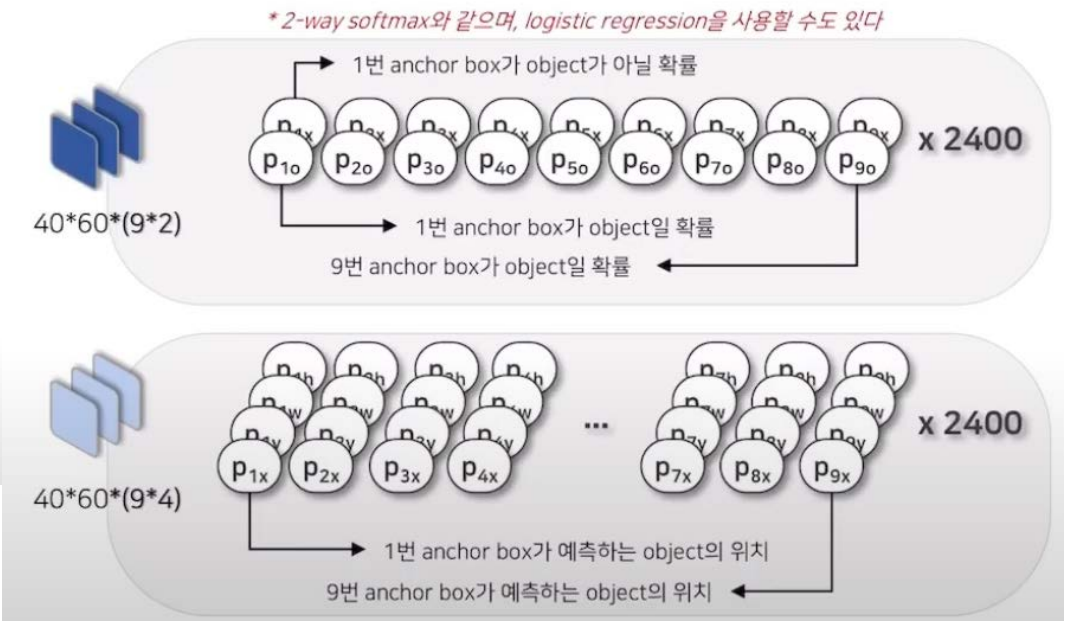




Prediction Network

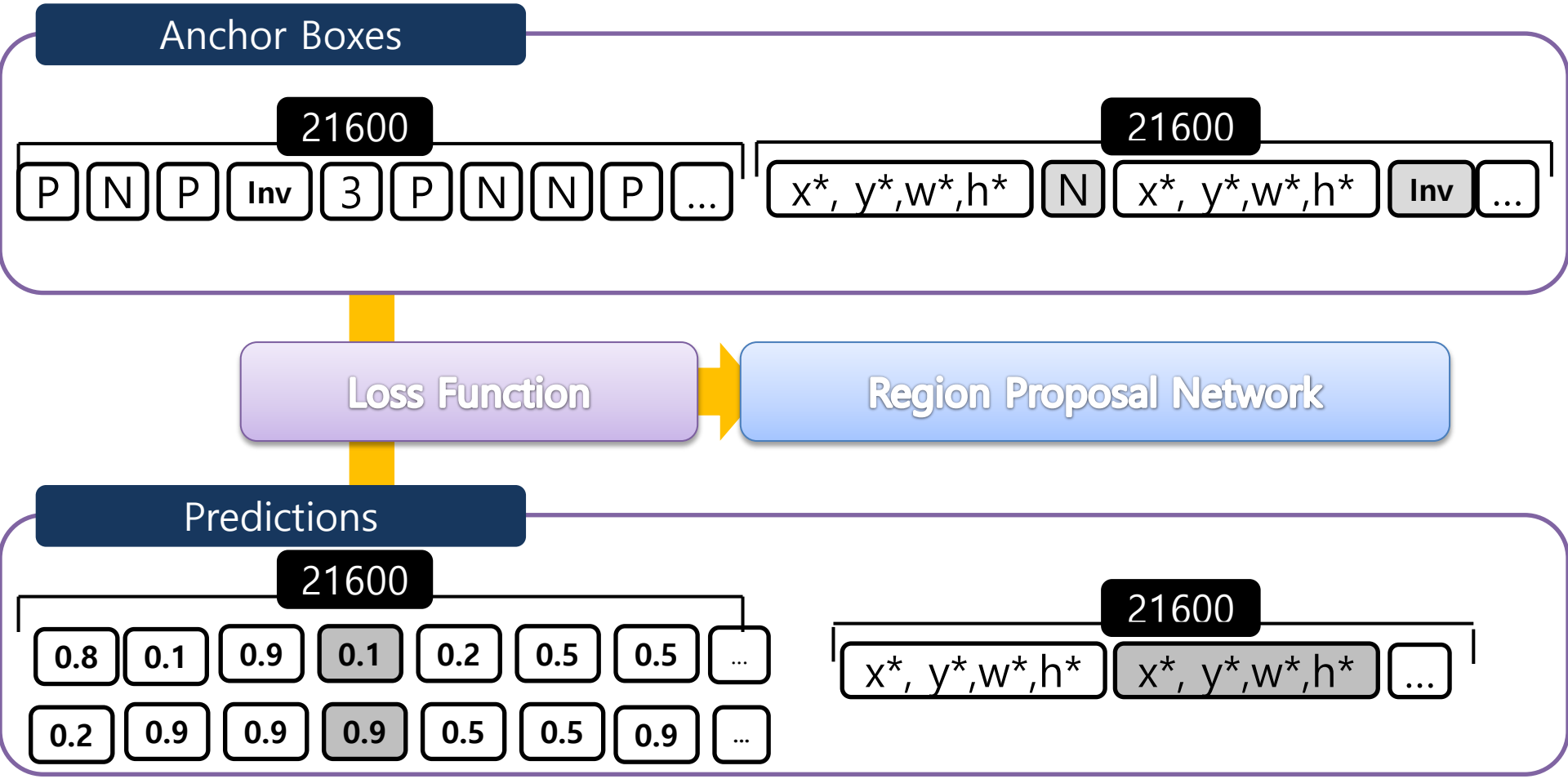
- 3*3 sliding window를 거친 feature map에 1*1 convolution kernel을 두 가지 목적(reg, cls)을 위해 적용
- 모든 40*60의 위치에 대해 classification과 bbox regression predicaiton 목적의 vector 생성 ->anchor box로 만든 vector와 train





Anchor & Prediction Network

- 각 anchor별로 9개를 예측한 결과(predictions)와, 동시에 anchor box로 생성한 vector 를 이용해 RPN train



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

Loss Function of RPN

- 각 anchor 별로 9개를 예측한 결과 (predictions)와, 동시에 anchor box로 생성한 vector를 이용해 RPN train

$$L(\{p_i\},\{t_i\})$$

$$\frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*)$$

하나의 anchor box가 object이면 pi=1, 아니면 0 → Log Loss 계산해 학습
softmax로 나타나는 각 벡터의 object 또는 background일 확률(Likelihood)을 곱하여 -log를 취한 형태 (값이 작을수록 loss가 적다)

+

$$\lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

p_i^*가 곱해지는 형태이기 때문에, 해당하는 anchor box에 object가 없다면 regressor는 학습되지 않음

$$L_{reg}(t_i, t_i^*) = SmoothL_1((\boldsymbol{x}-\boldsymbol{x_a})/\boldsymbol{w_a} - (\boldsymbol{x^*}-\boldsymbol{x_a})/\boldsymbol{w_a}))$$

위 Loss에서 줄여야 되는 부분은 결국 $(x - x^*)$이며, 결과적으로는 anchor를 매개체로 사용하여 ground truth와 최대한 가까운 proposal을 생성(예측)하도록 학습된다고 볼 수 있음

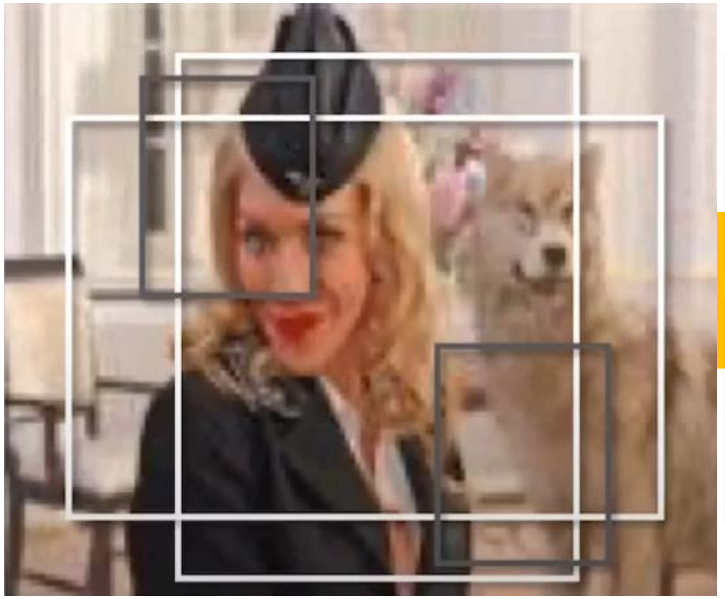
$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$$

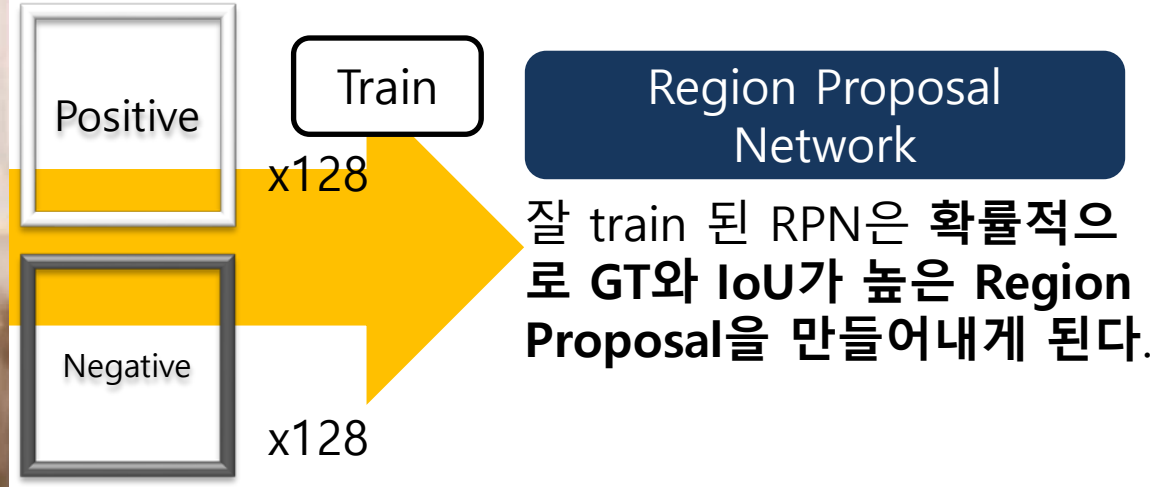
$$t_x = (x - x_a)/w_a, \quad t_y = (y - y_a)/h_a,$$
$$t_w = \log(w/w_a), \quad t_h = \log(h/h_a),$$
$$t_x^* = (x^* - x_a)/w_a, \quad t_y^* = (y^* - y_a)/h_a,$$
$$t_w^* = \log(w^*/w_a), \quad t_h^* = \log(h^*/h_a),$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

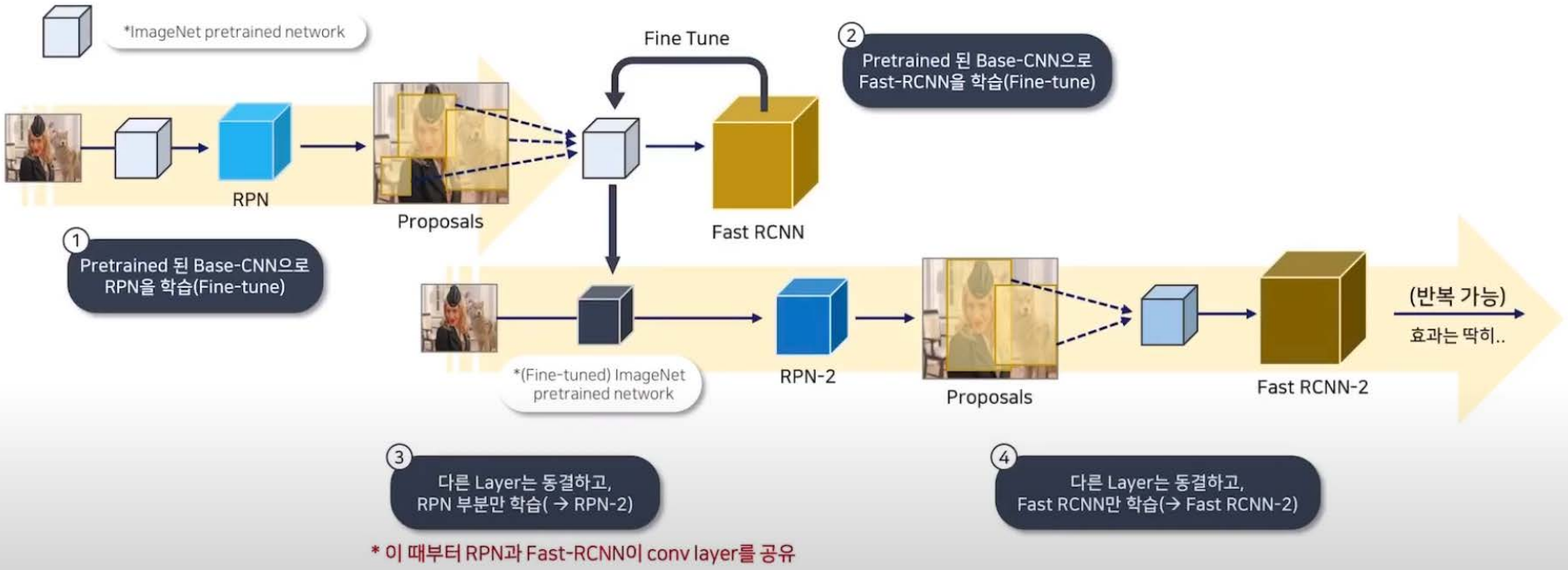
Using mini-batch in training

- RPN train시 “image-centric” sampling을 사용
- 하나의 image에서 256개의 anchor box를 sample하되, positive와 negative 비율이 (최대한) 1:1이 되도록 sample
  (Positive보다 Negative에 치우쳐(biased) 학습이 될 가능성을 최소화)



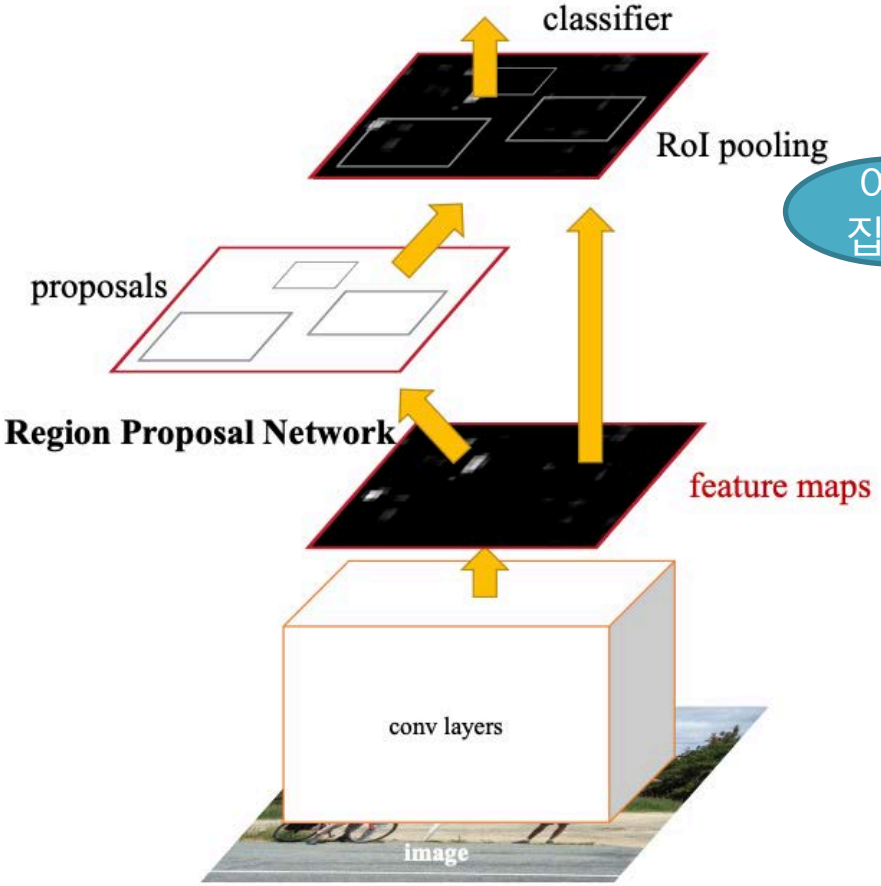


4-step Alternating Training

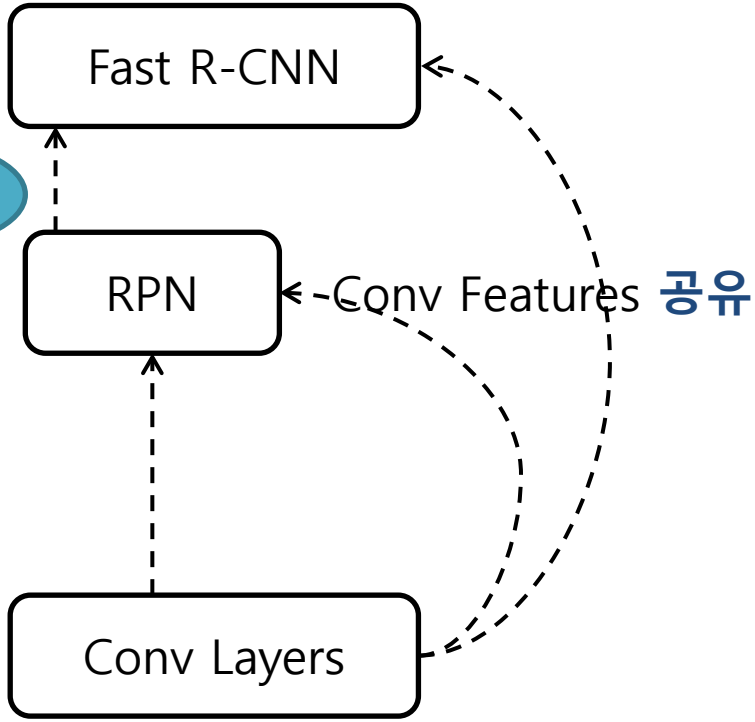


→ 결과적으로 RPN과 Fast RCNN이 Convolutional Feature를 공유할 수 있도록 하는 학습

RPN+Fast R-CNN

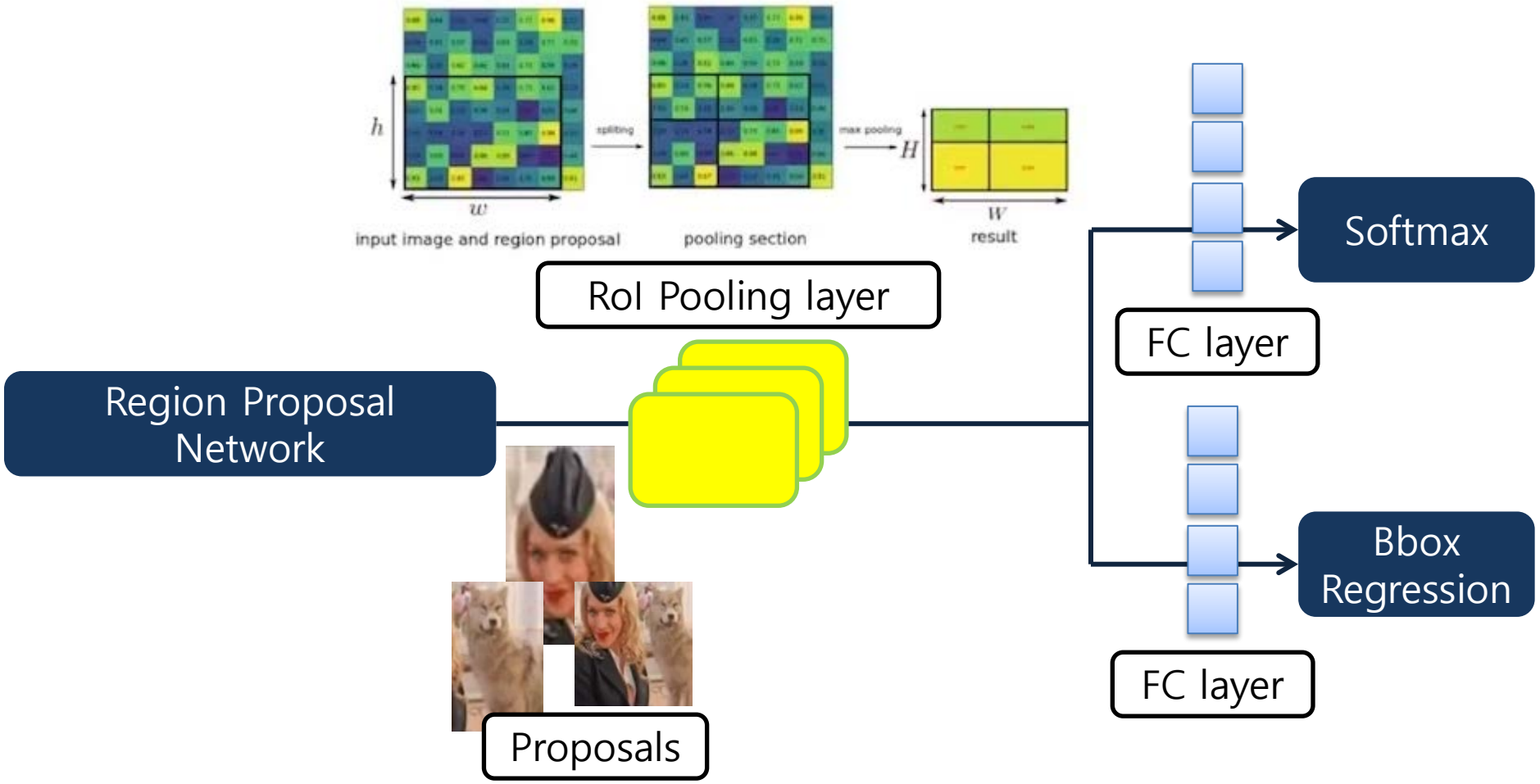


여길 집중!!

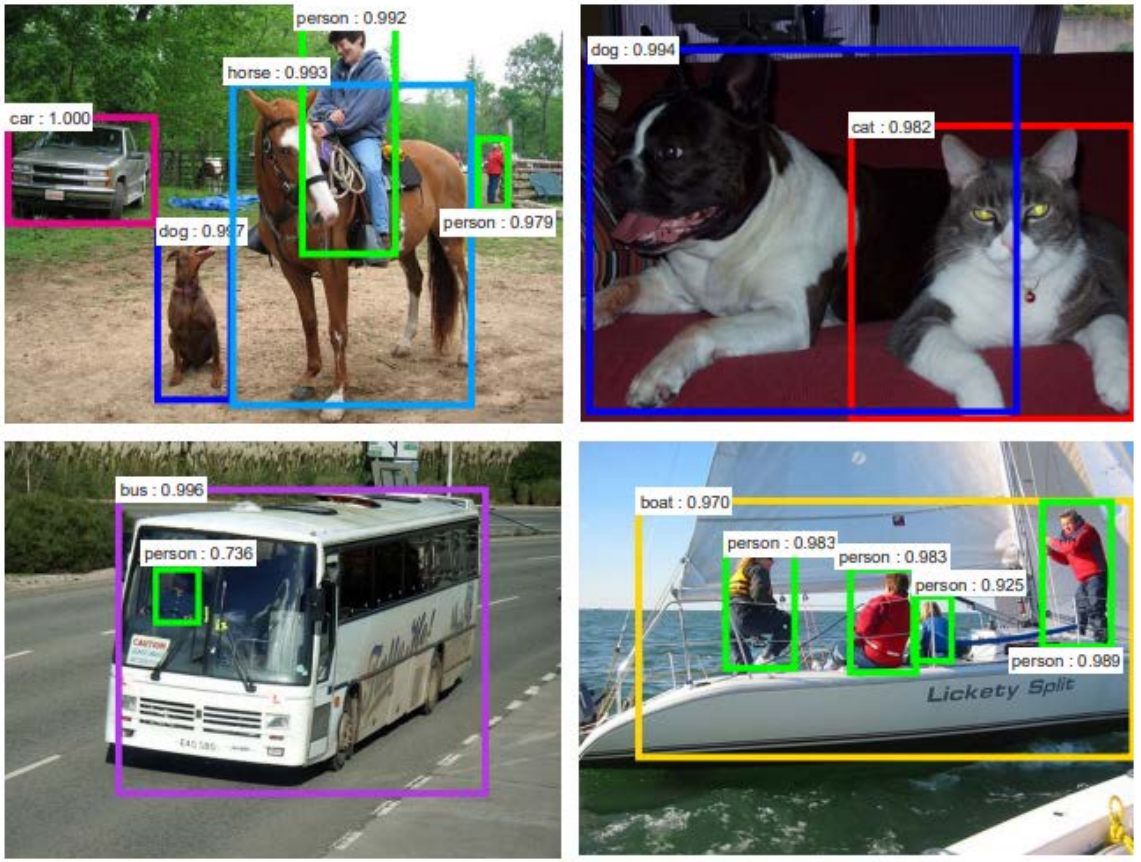


RPN+Fast R-CNN

- Train된 RPN에서 뱉는 propsal은 Fast RCNN으로 넘어가 classification 및 regression을 수행



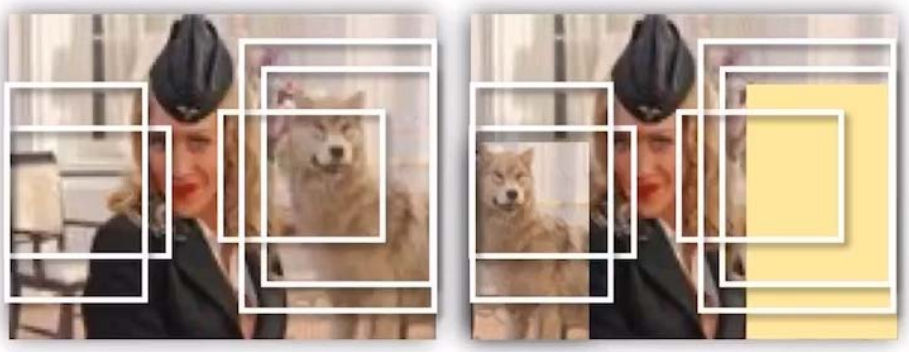
Anchor Boxes are important



큰 object에 대해서도 준수한 예측성능을 보이고
이미지를 벗어나는 object에 대해서도 예측 가능

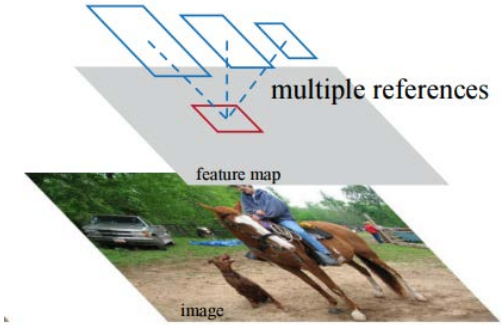
Anchor Boxes are important

Translation Invariant



- ✓ Anchor를 sliding 방식으로 사용하면서, 물체의 translation에 invariant한 성질
- ✓ MultiBox는 k-means를 통해 anchor를 생성(translation variant)함. 이 방식에 비해 Faster-RCNN은 훨씬 적은 parameter를 갖게 되며,동시에 overfitting 확률도 낮춤

Multi-scale Anchors as Regression References



- ✓ Multi-scale prediction을 위해 pyramid-of-image, pyramid-of-filters등 방법 존재
  (각각 다른 image,필터 scale에 의존)
- ✓ Faster RCNN이 사용하는 pyramid of anchors방식은 동일한 scale의 feature map과 동일한 크기의 kernel에 의존
  ->feature sharing의 key component

train-time region proposals method	# boxes	test-time region proposals 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

method	proposals	training data	COCO val mAP@.5	COCO val mAP@[.5, .95]	COCO test-dev mAP@.5	COCO test-dev 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**

On MS COCO

Pascal VOC 2007 bench mark

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**

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**

Is VGG-16 Better?

	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 stage vs Two stage

Conclusion

- 효율성
  - Convolutional feature를 Fast R-CNN과 공유함으로써 region proposal이 near cost free
- 정확도
  - Detector로 fine-tuning된 RPN을 이용하게 되면서, 좀더 정확한 localization이 가능한 detection system 구축

Thank You