R-CNN : CNN 사용 최초의 object detection 방법

* 단점 1. AlexNet 사용 224×224 강제 warping ⇒ 성능↓

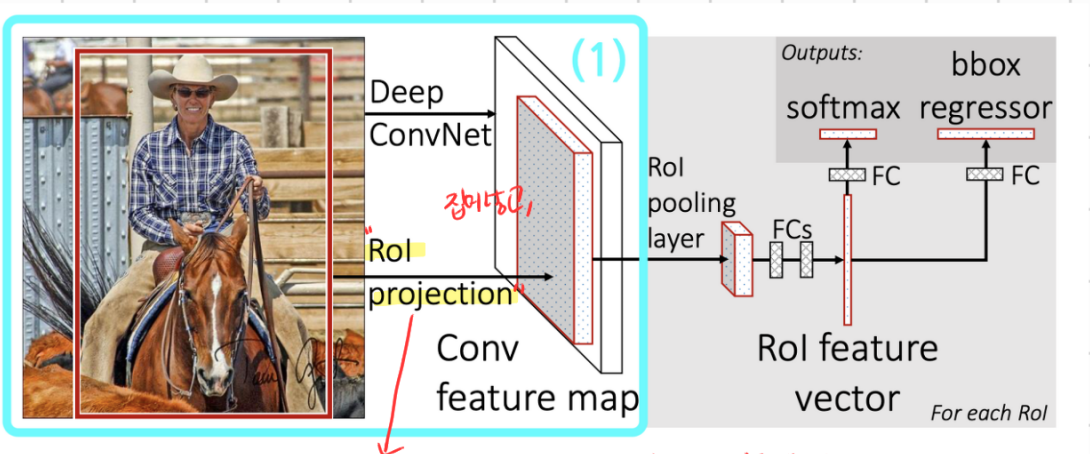
2. Selective search 통해 2000개 image proposal all input to CNN ⇒ time↑

3. 〃 나 SVM이 GPU 사용 장점 구조 ×

4 뒤 부분 수행 Computation share × (No back propagation)

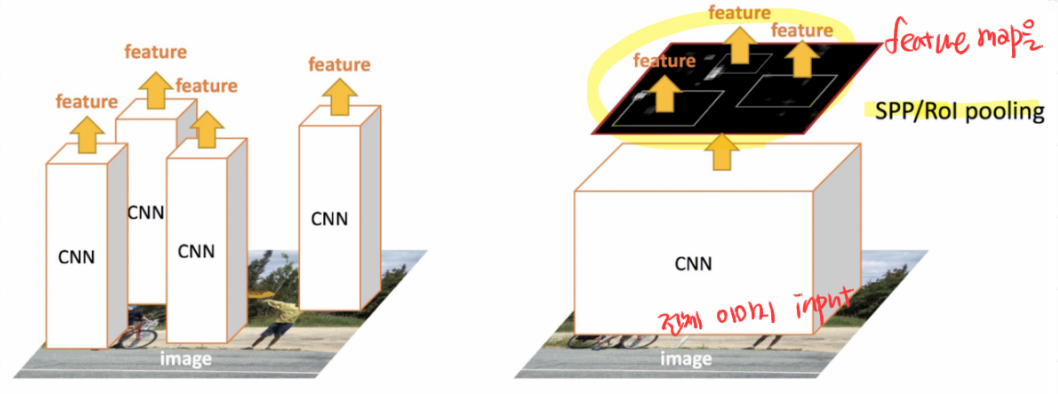
Fast R-CNN

1. CNN model 변형



- RoI region of interest

selective search로, 미리만 따져 좌표 받음



R-CNN

- Extract image regions
- 1 CNN per region (2000 CNNs)
- Classify region-based features

SPP-net & Fast R-CNN (the same forward pipeline)

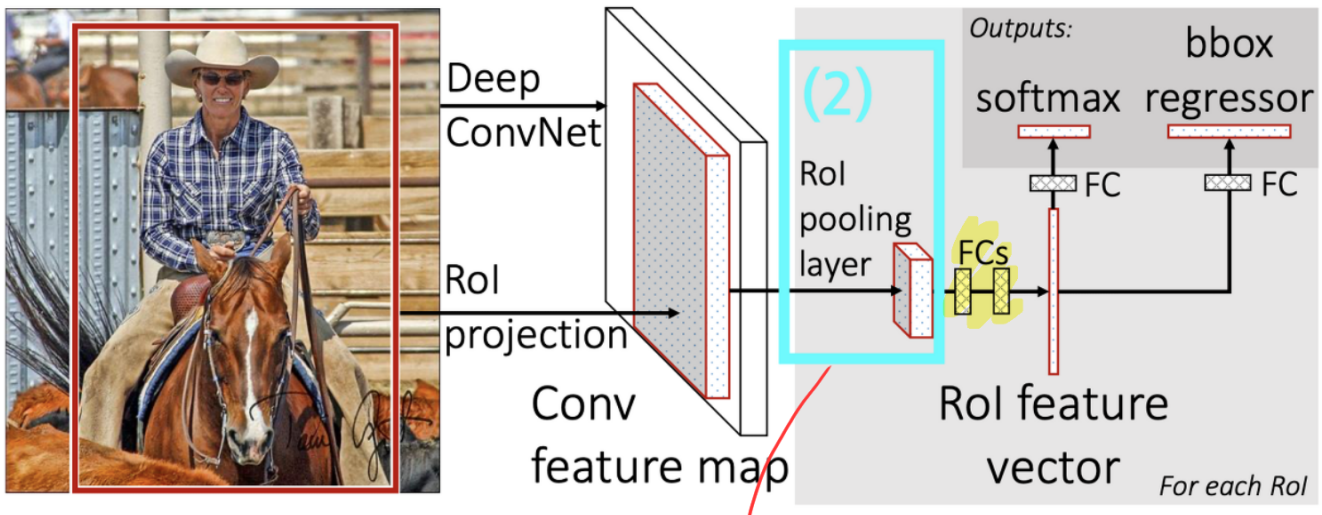
- 1 CNN on the entire image
- Extract features from feature map regions
- Classify region-based features

- 붉은색 내역을 crop

(투영)

- crop × 전체이미지를 CNN model에 강제넣은후 feature map에 RoI Projection

2. RoI Pooling



→ 같은 resolution (해상도)의 feature map 필요

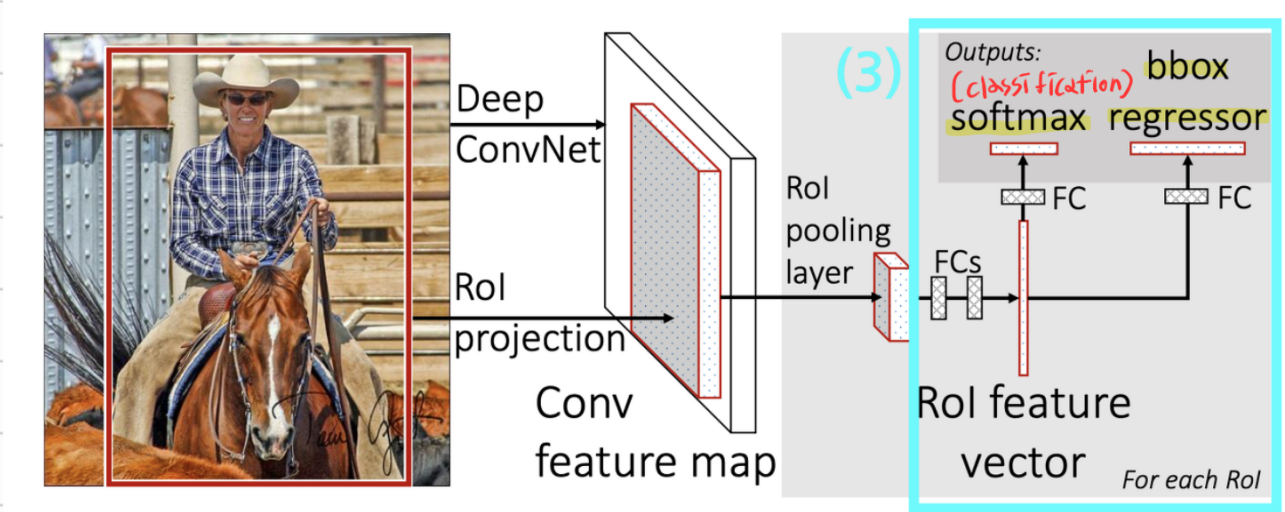
- Resolution의 크기를 맞추기 위해 RoI pooling 수행
  - → 크기 ≠ feature map의 셀마다 max pooling

ex) n × n output feature map. = fixed length feature vector n × n

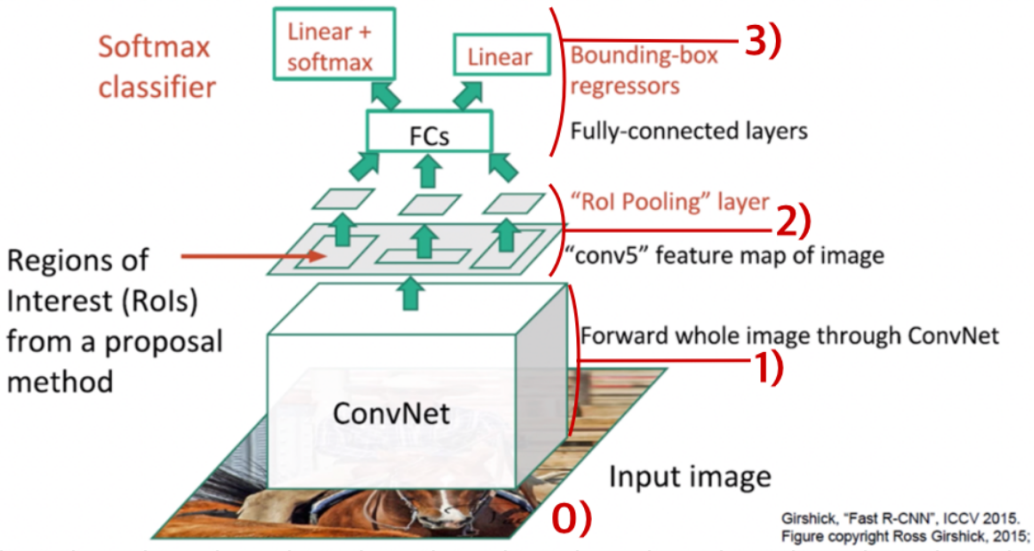
21 × 14 → 21/3 = 7, 14/2 = 7 stride (3,2)

42 × 35 → 42/6 = 7, 35/5 = 7 " (6,5)

3. Classification & Bounding Box Regression



* Fast R-CNN 정리



0) Input Image에 Selective search 진행, RoI 영역 미리뽑고

1) CNN 모델에 Image 전체 Input, Feature map output

2) 뽑아놓은 RoI 영역을 Feature Map에 적용후, RoI pooling 해서 fixed Length feature vector output

3) FCL을 거쳐 두개의 Layer로 class prediction, Bounding Box Regression 진행

L Loss function

동시학습

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v)$$

Classification

$p = (p_0, \ldots, p_K)$: 예측된 Class Score in $K+1$ categories (0은 background)

$u =$ 실제 Class Score

Log loss

Bounding box Regression

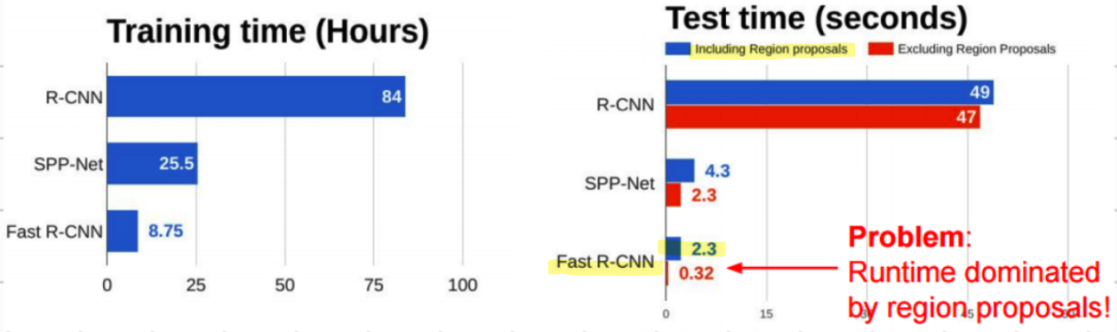
$t^u = (t^u_{\text{x}}, t^u_{\text{y}}, t^u_{\text{w}}, t^u_{\text{h}})$: 예측된 tuple

$(v_{\text{x}}, v_{\text{y}}, v_{\text{w}}, v_{\text{h}})$: 실제 Bounding box 좌표 값

Smooth L1 Loss

u가 0(background)일때 0, 나머지일때 1

<문제>



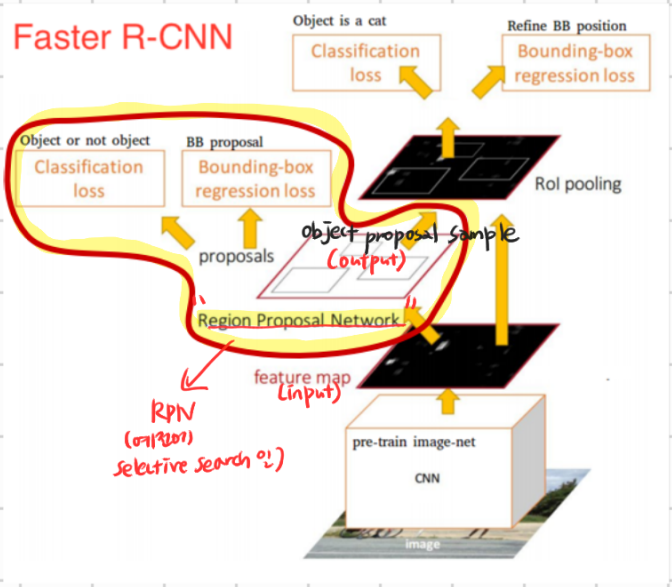
Not real-time object detator

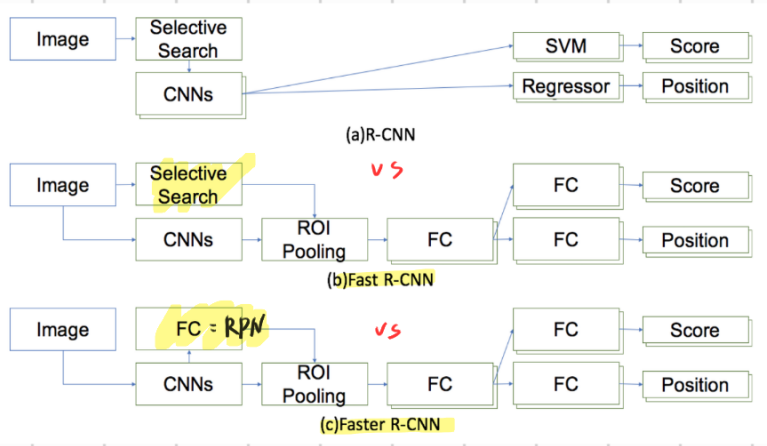
- Region Proposal에 사용되는 selective search가 GPU에서 CPU 사용 알고리즘 때문

  ⇒ Faster R-CNN에서 해결됨

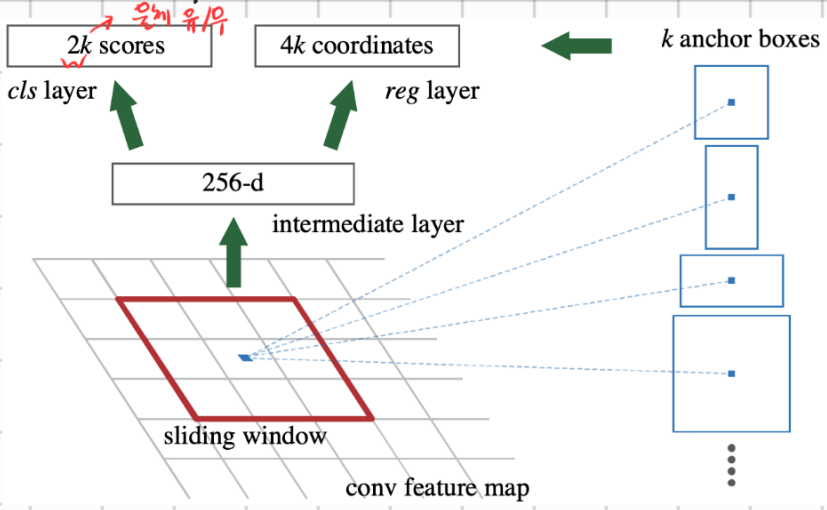
*Faster R-CNN

- Deep network 기반 Region Proposal 알고리즘 = RPN (Region Proposal Networks)





1. RPN (Region proposal Network)

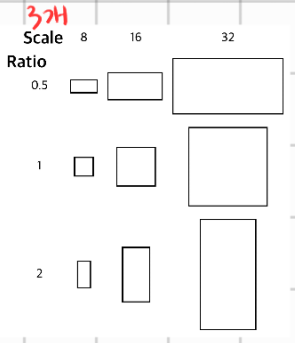


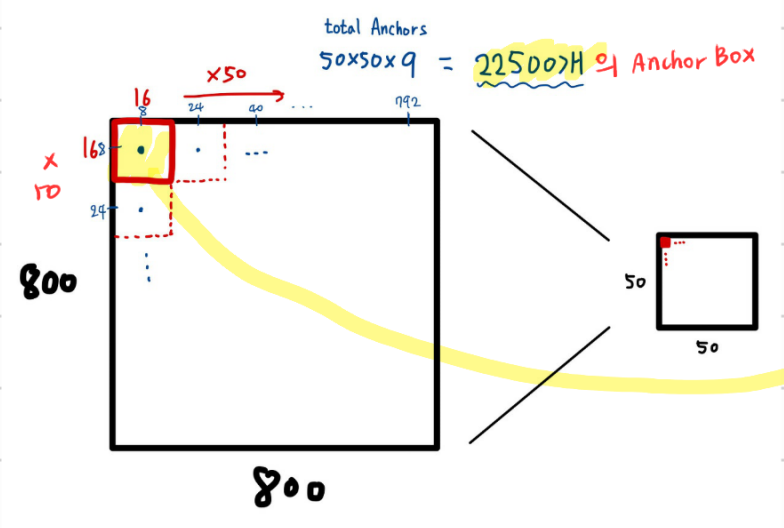
VGG-16을 백본 (backbone)로 사용

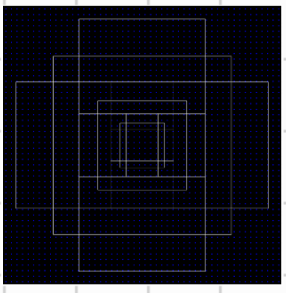
1) Anchor targeting

800×800×3 → CNN (VGG-16) → 50×50×512 feature map
(Input)

→ sliding window 방법 → k=9개 Anchor Box 생성

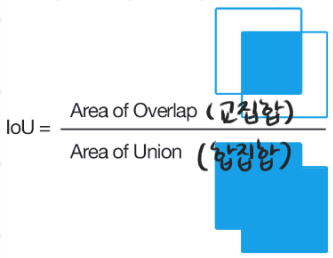






9개 Anchor Box

- 22500개의 Anchor Box 기준 detection 학습! 위해 Labeling 필요
- (↳ Ground Truth)
- GT Label은 22,500개의 Anchor & Ground Truth Box의 IOU를 모두 계산

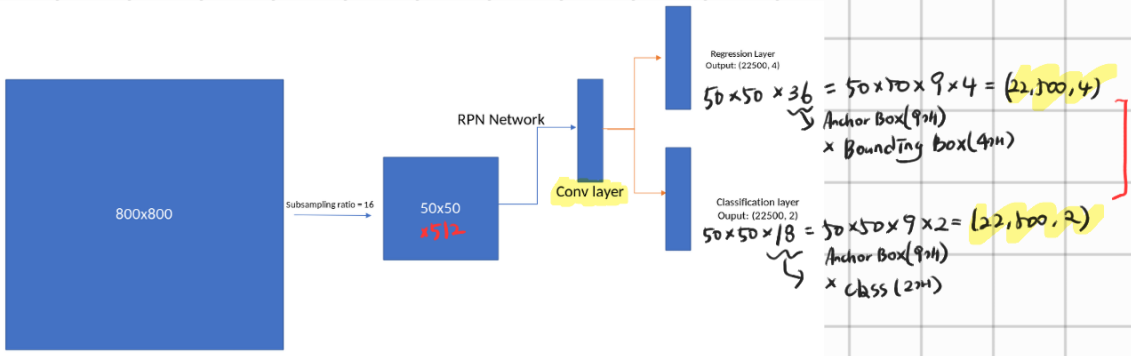


0.7 ↑ = 1 = positive

other = -1 신경 X

0.3 ↓ = 0 = Negative

2. prediction



1) prediction 값으로 GT Label 과함께 Loss function을 통해 RPN 학습
↳ P:N = 1:1 비율로 Sampling

&

2) prediction 값은 NMS거쳐 RoI sampling 된 후 Fast-RCNN의 입력 like selective search
↳ Non-maximum Suppression
: Detection에서 중복되는 Bounding Box제거

3. Loss function for RPN

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

Log Loss
GT Label (물체가 있으면 1, 없으면 0)
물체가 없으면 bbox는 고려하지 않음.
Smooth L1 Loss
GT box
Predicted Probability of Anchor i

4. 성능

	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**