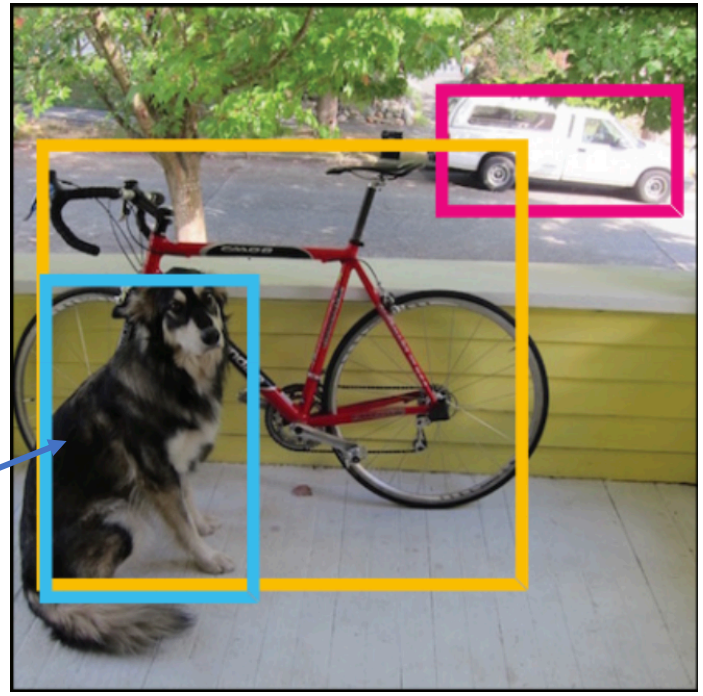
Object Detection

Object Detection Task 의 구성

Classificaiton + Localization

Calas : Dog
Box : (x, y, w, h)



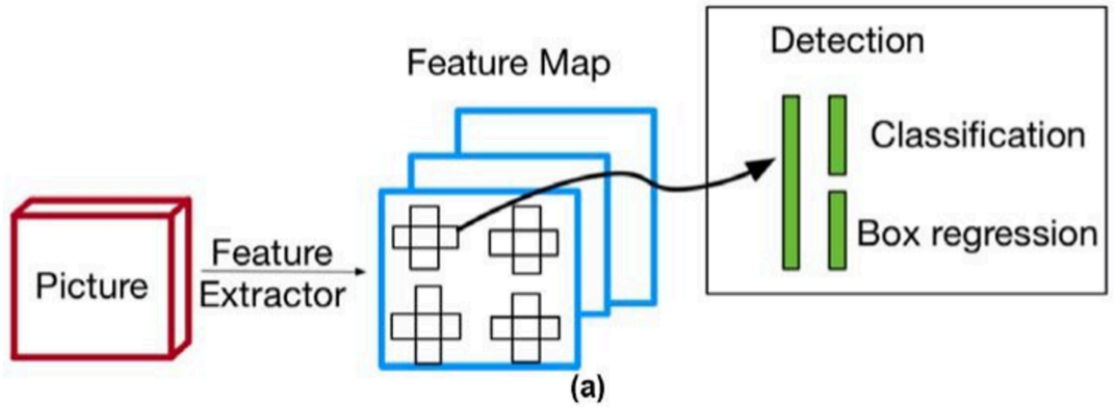
Single Stage Detector, Two Stage Detector

Compare

Single Stage Detector

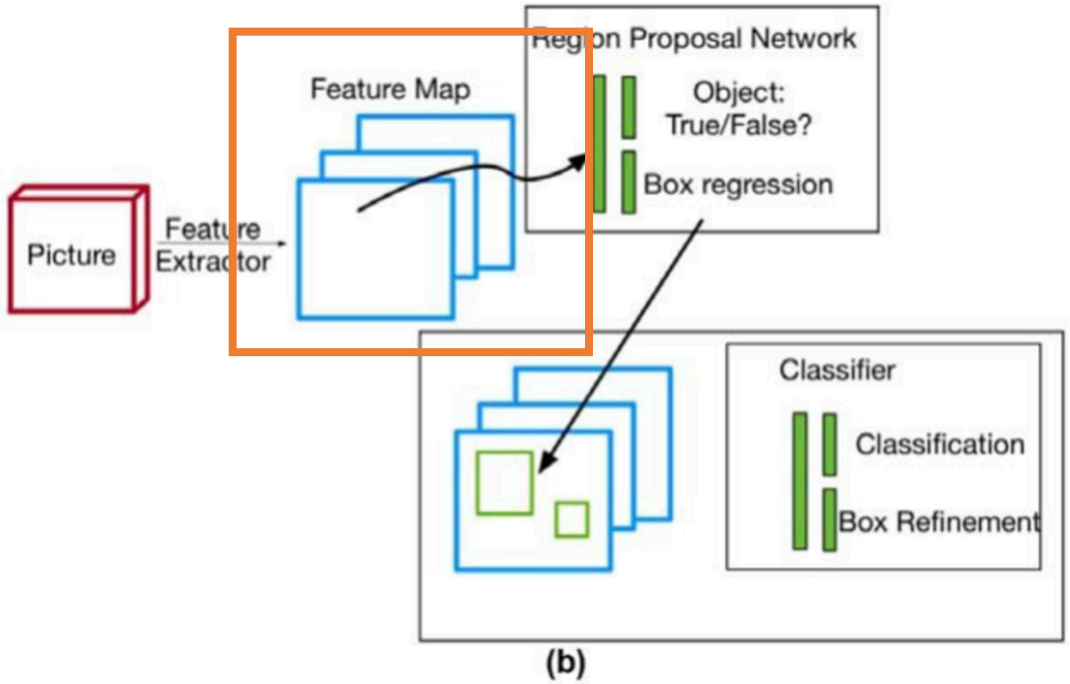
No explicit RoI pooling (Region proposal)

RoI를 찾아내는 별도의 layer, algorithm이 없음



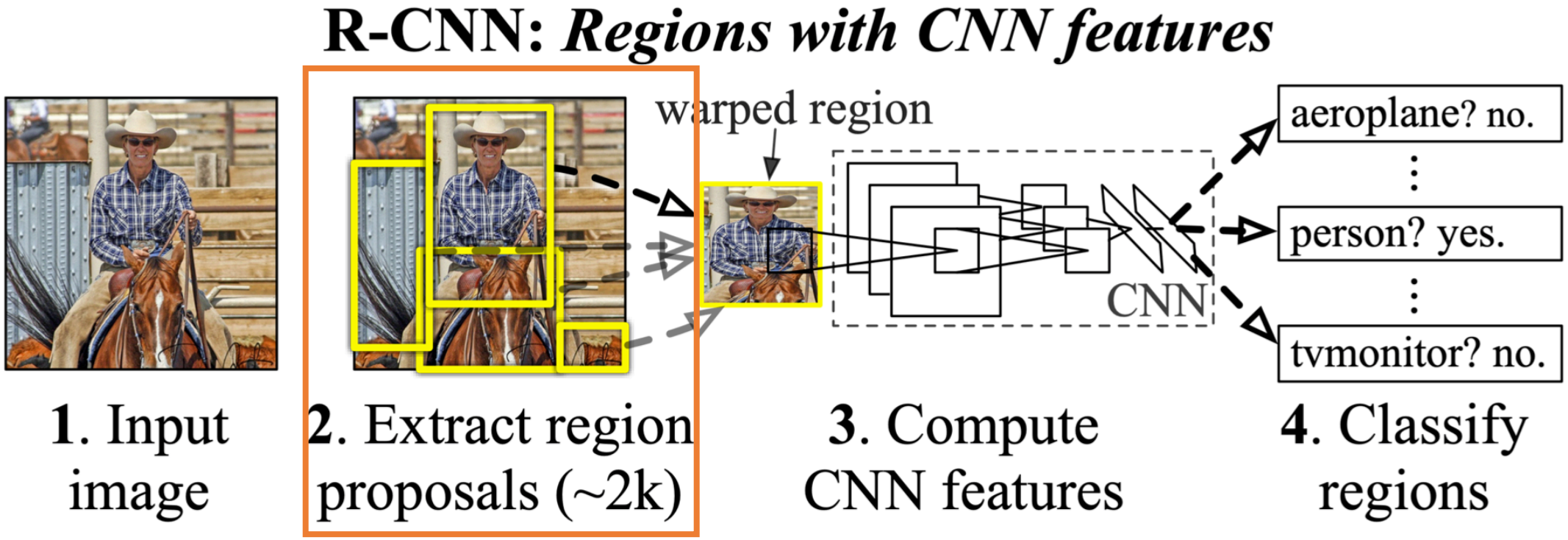
YOLO, SSD

Two Stage Detector



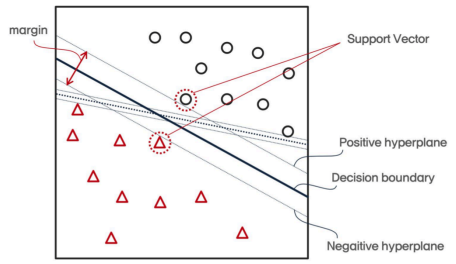
RCNN, Fast RCNN, Faster RCNN

RCNN



Region proposal

SVM



Selective Search Selective Search for Object Recognition - J.R.R. Uijlings ..(IJCV 2012)

Hierarchical Grouping Algorithm

Selective Search Selective Search for Object Recognition - J.R.R. Uijlings ..(IJCV 2012)

Hierarchical Grouping Algorithm

$R : \{ r_1, r_2, \dots \}$ 선택된 후보 집합들

$S : \{ s(r_i, r_j), \dots \}$ 선택된 후보 집합들

1. 초기 sub-segmentation 을 수행

2. 작은 영역을 반복적으로 큰 영역으로 통합

3. 통합된 영역을 바탕으로 후보 영역을 만들어냄

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
  Calculate similarity $s(r_i, r_j)$
  $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
  Get highest similarity $s(r_i, r_j) = \max(S)$
  Merge corresponding regions $r_t = r_i \cup r_j$
  Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
  Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
  Calculate similarity set S_t between r_t and its neighbours
  $S = S \cup S_t$
  $R = R \cup r_t$

Extract object location boxes L from all regions in R

https://donghwa-kim.github.io/SelectiveSearch.html

Selective Search

Selective Search for Object Recognition - J.R.R. Uijlings ..(IJCV 2012)

Hierarchical Grouping Algorithm

1. 초기 sub-segmentation 을 수행

Efficient Graph-Based Image Segmentation

Vertices : pixcel

Edge : 두 픽셀간의 차이에 대한 정도를 나타낸 값으로 음수가 아닌 값을 사용하는데, 인텐시티나 컬러, 모션, 위치에 대한 차이

2. 작은 영역을 반복적으로 큰 영역으로 통합

두 영역간의 유사도를 측정

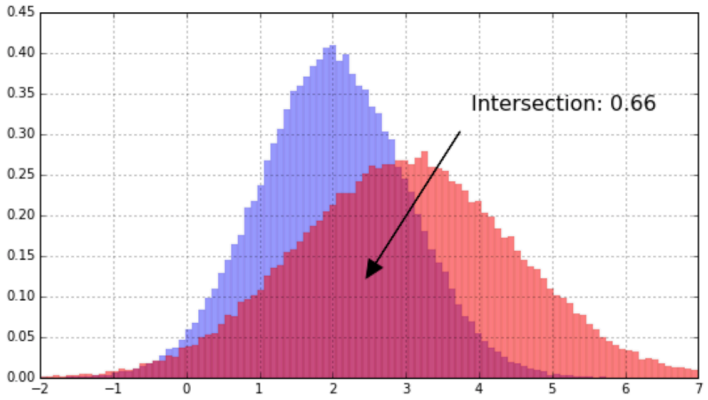
유사도의 측정 기준 : 정규화된 Color, Texture, Size, Fill 의 가중합

3. 통합된 영역을 바탕으로 후보 영역을 만들어냄

Region Proposal

Selective Search

- Color: 이미지의 색깔
  - 각 컬러 채널을 25개 bin으로 설정
  - 각 region 마다 컬러 히스토그램 생성 $C_i = c_i^1, \ldots, c_i^n$
  - 차원 수 n = 75 (RGB 3채널 × 25개의 Bin)
  - L_1 norm 정규화 $[0, 1]$
  - 인접한 regions의 교집합을 유사도로 측정
  - $s_{colour}(r_i, r_j) = \sum_{k=1}^{n} min(c_i^k, c_j^k)$
  - 유사성 척도의 min function은 두 히스토그램의 교집합을 나타냄



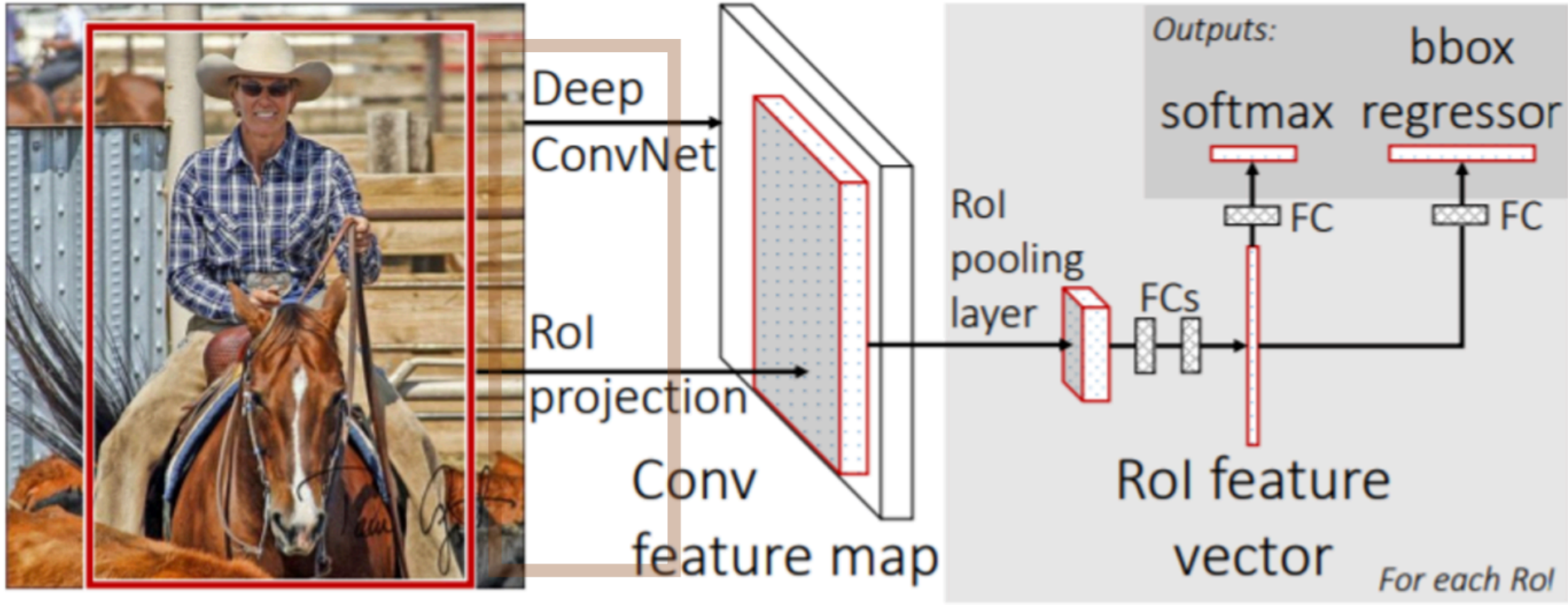
- Texture: 주변 Pixel값 들의 변화량 [참고자료: HoG]
  - $\sigma = 1$인 8방향의 가우시안 미분을 적용
  - 10개의 Bin으로 히스토그램 도출 $T_i = t_i^1, \ldots, t_i^n$
  - L_1 norm 정규화 $[0, 1]$
  - 80차원(8방향 × 10차원의 Bin)의 벡터로 인접한 region들의 유사성을 평가
  - RGB(3차원)의 경우: 8방향 × Bin의 수(10차원) × RGB(3차원) = 240차원
  - $s_{texture}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k)$

- Size: Region들의 사이즈
  - 사이즈가 작을 수록 유사도가 높음
  - $s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}$
  - im은 원 이미지를 나타냄

- Fill: candidate Bounding Box 크기와의 차이
  - candidate Bounding Box와 Region들의 사이즈의 차이가 적을수록 유사도가 높음
  - $s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}$
  - im은 원 이미지를 나타냄

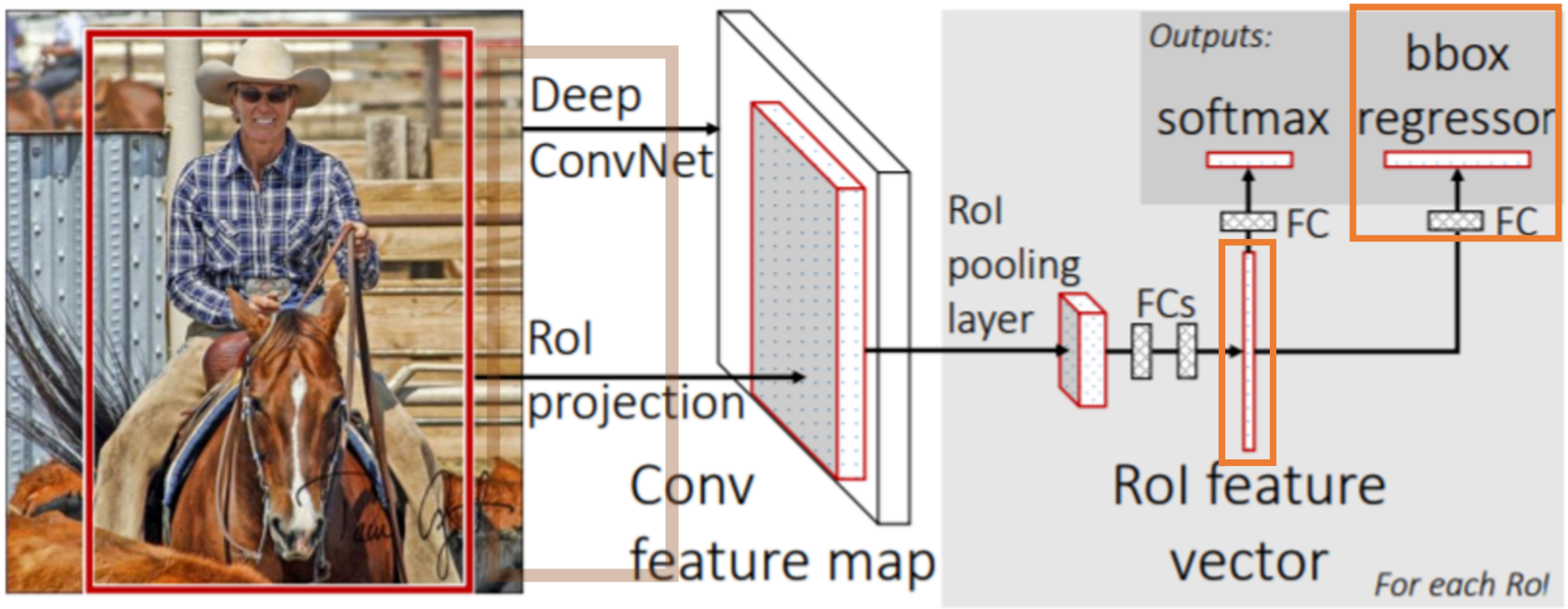
https://donghwa-kim.github.io/SelectiveSearch.html

Fast RCNN



Region proposal

Fast RCNN Region proposal 을 통해서 받아진 projection 에서 RoI를 잘 찾을 수 있게 학습

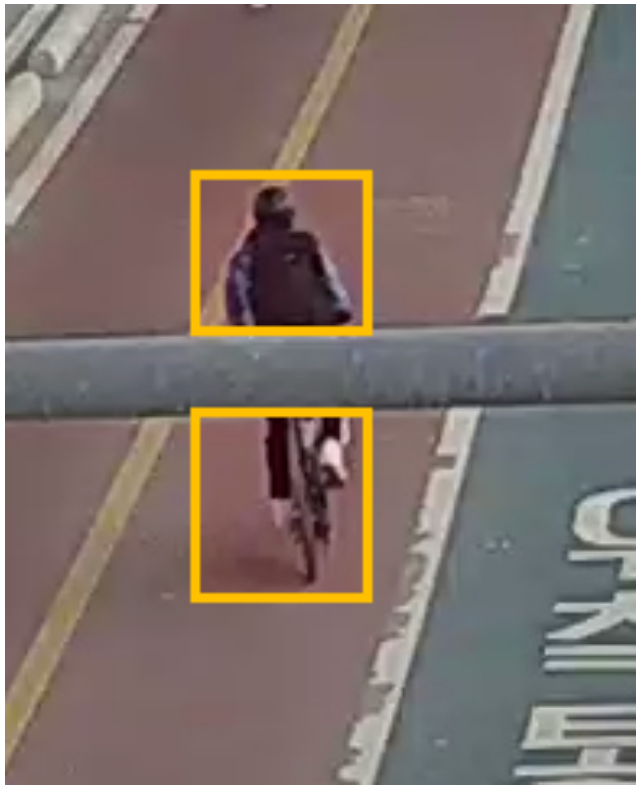


Region proposal

Region proposal 로 받은 모든 box를 연산하는 것이 아닌 의미 있다고 판단한 box만을 추출

What is difference?

RCNN



Fast RCNN

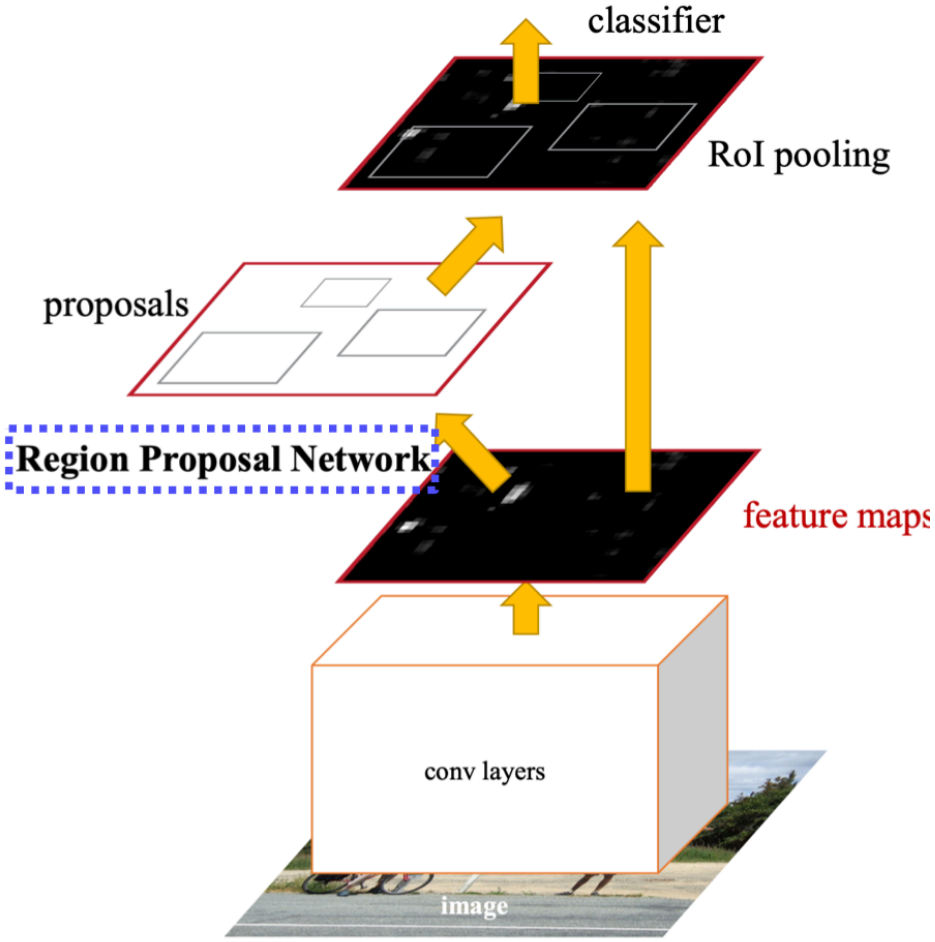
Faster RCNN Region proposal 도 학습 가능한 구조로 변경

RPN (Region Proposal Network)

Anchor Box

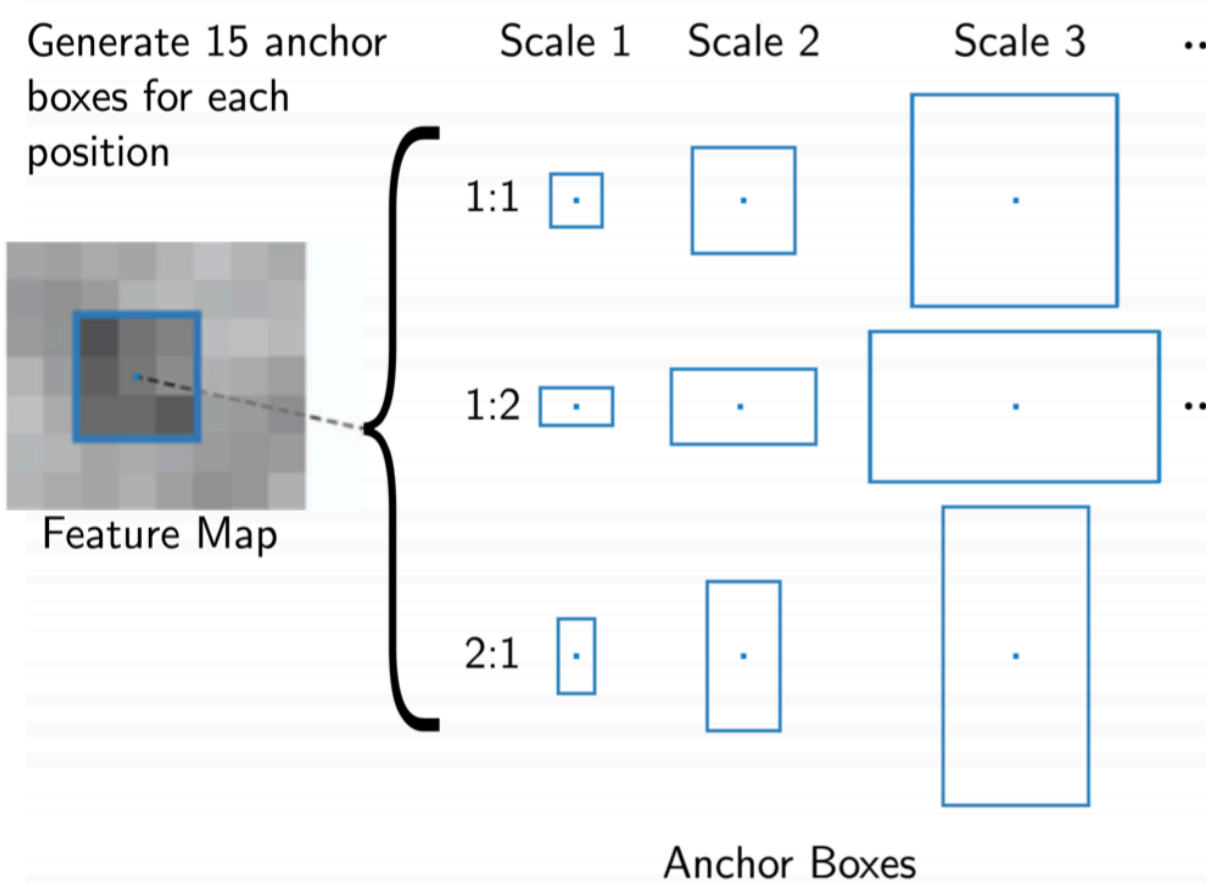
Region proposal Network 학습을 위한
Box들의 layout

**일단 개괄적인 bounding box에 대한 정의를 해두고
디테일 하게 학습하게 하자! 라는 취지**

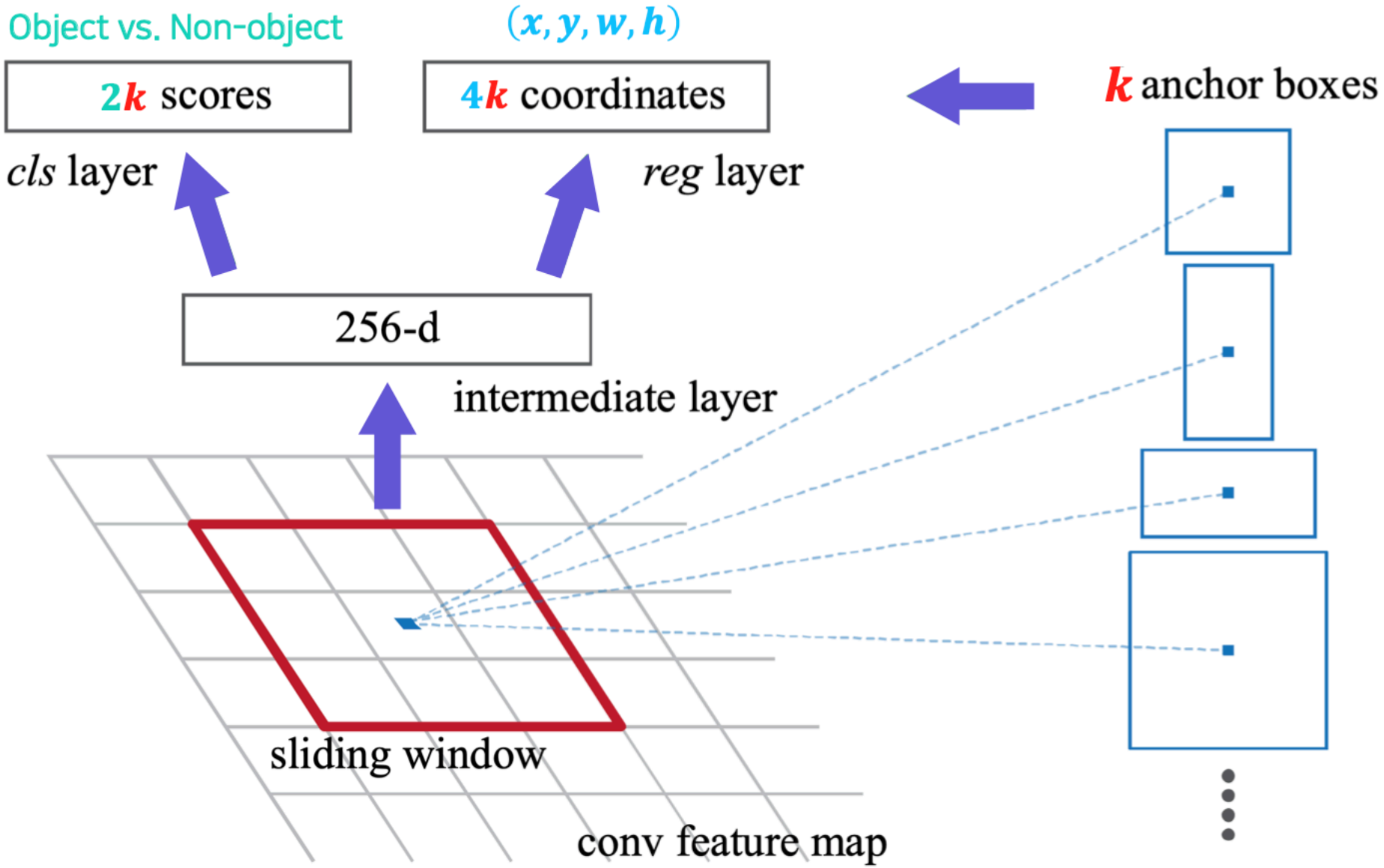


Anchor Box

Hyper parameter



Faster RCNN


Object vs. Non-object
(x, y, w, h)
2k scores
4k coordinates
k anchor boxes
cls layer
reg layer
256-d
intermediate layer
sliding window
conv feature map

RPN을 이용한 Object Detection 의 output

예측된 모든 box들과 class confidence

Non-Maximum Suppression (NMS)

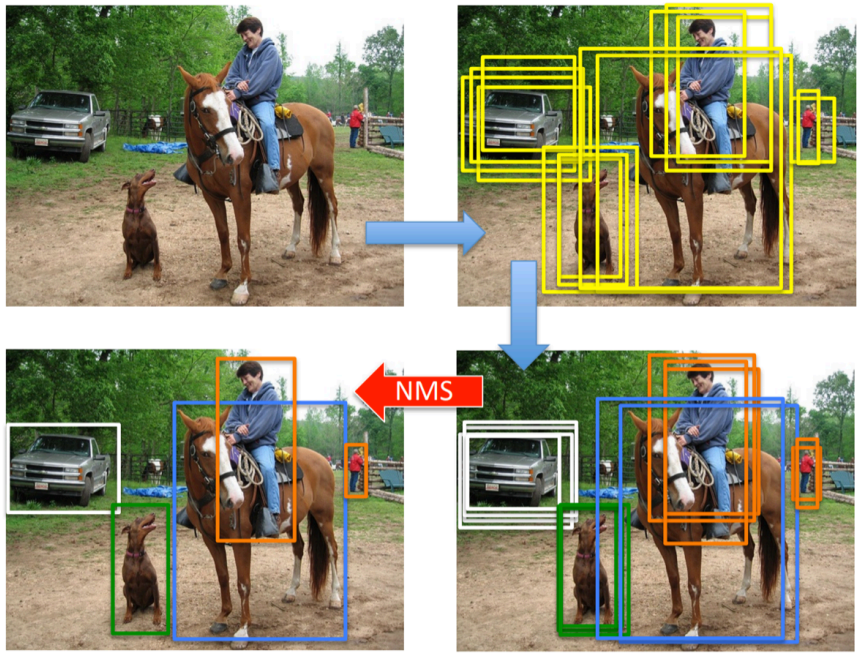
Step 1: Select the box with the highest objectiveness score

Step 2: Compare IoU of this box with other boxes

Step 3: Remove the bounding boxes with IoU ≥ 50%

Step 4: Move to the next highest objectiveness score

Step 5: Repeat steps 2-4



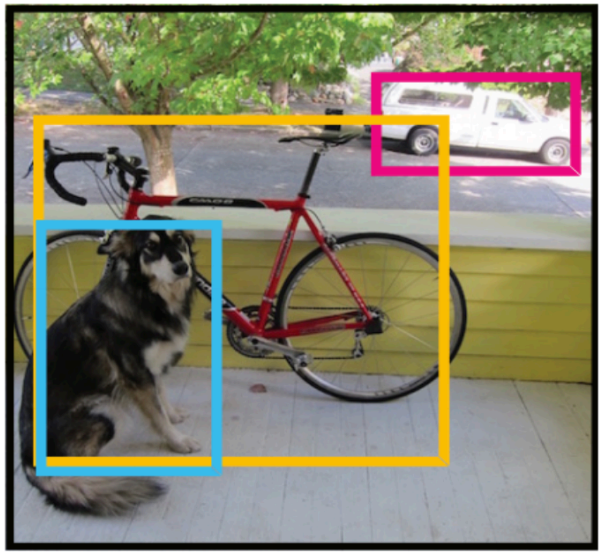
Soft-NMS- Improving Object Detection With One Line of Code, Navaneeth Bodia (ICCV 2017)

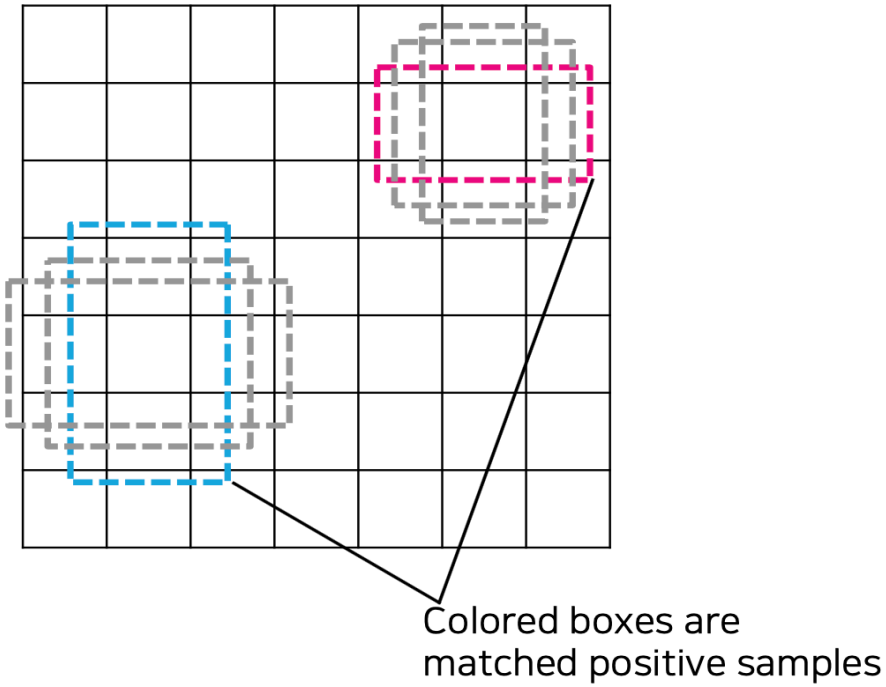
YOLO

1. **Single Stage Detection → (No RPN)**

2. Grid 로 나눠서 각 Grid(Feature map) 별 box, class 에 대한 정보 출력

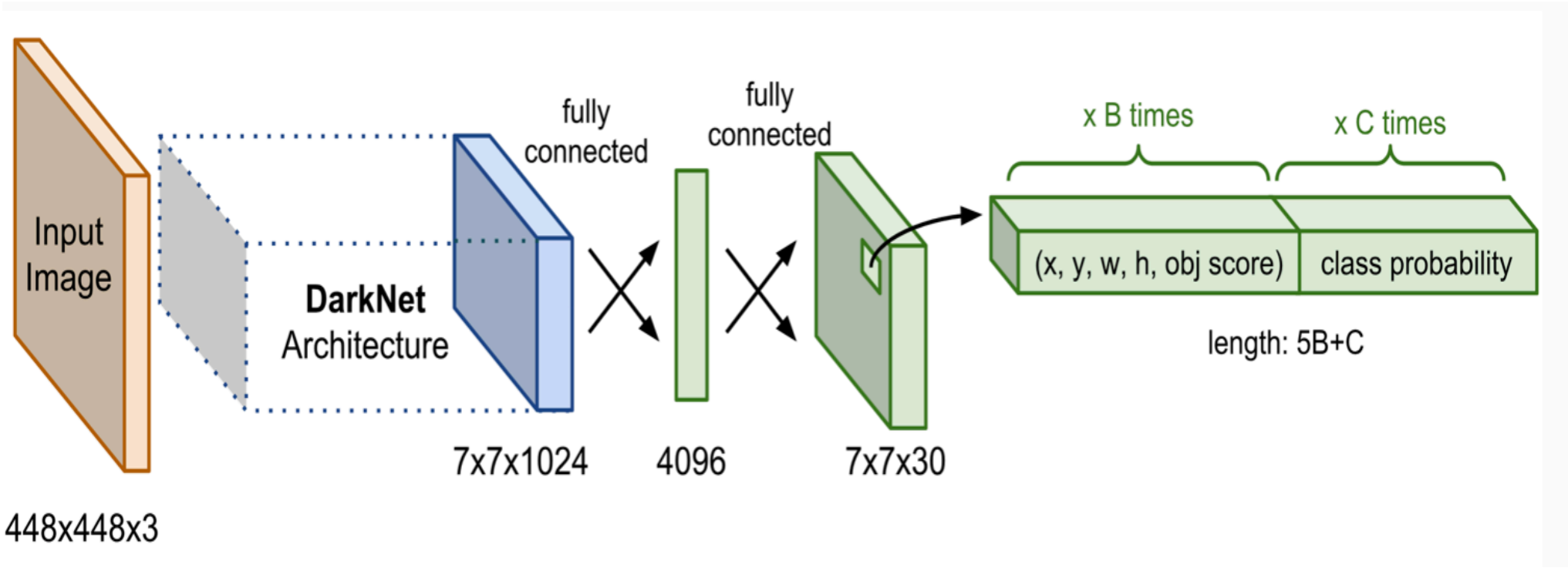
   - Input : image
   - Output : per Grid * Anchor box // box info(x, y, w, h) , class(one hot vector)
   - Target.: per Grid * Anchor box // box info(x, y, w, h) , class(one hot vector)





YOLO

유사 CNN

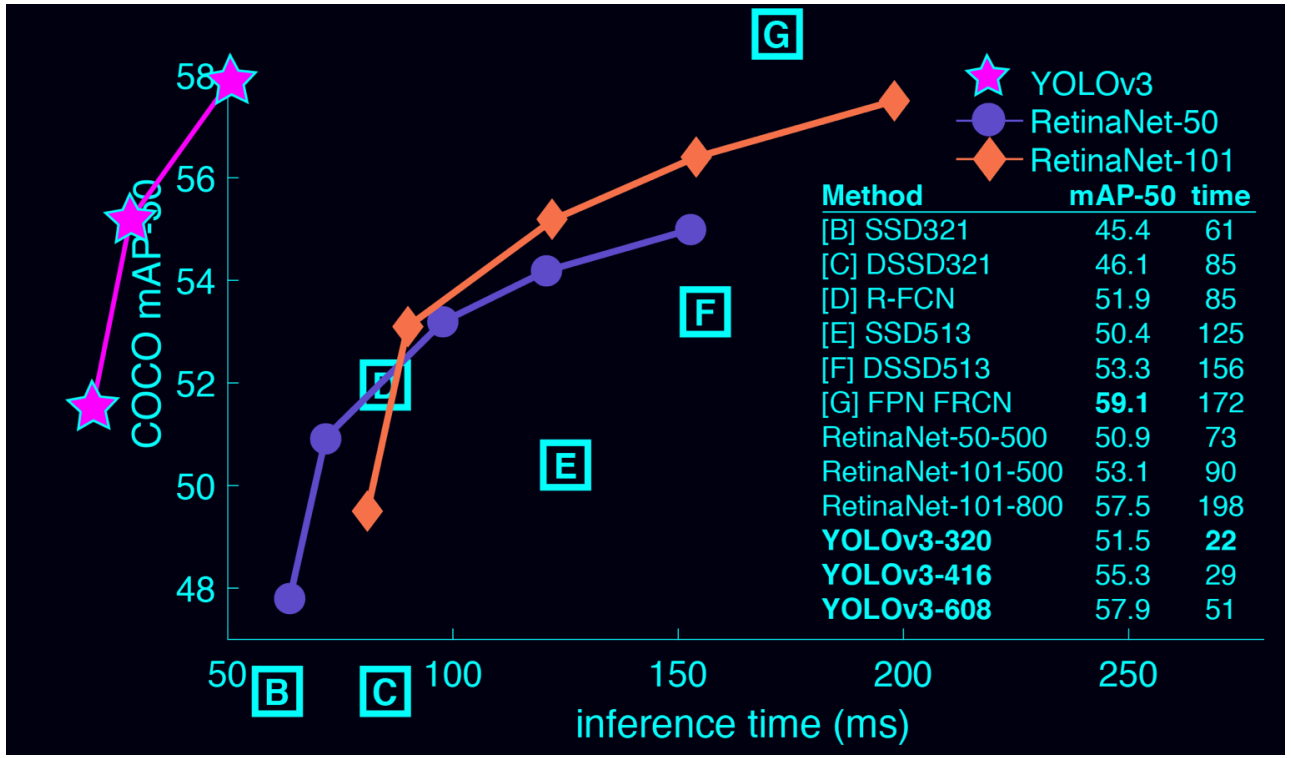

Input Image
448x448x3
DarkNet Architecture
7x7x1024
fully connected
4096
fully connected
7x7x30
x B times
x C times
(x, y, w, h, obj score)
class probability
length: 5B+C

YOLO

Version..

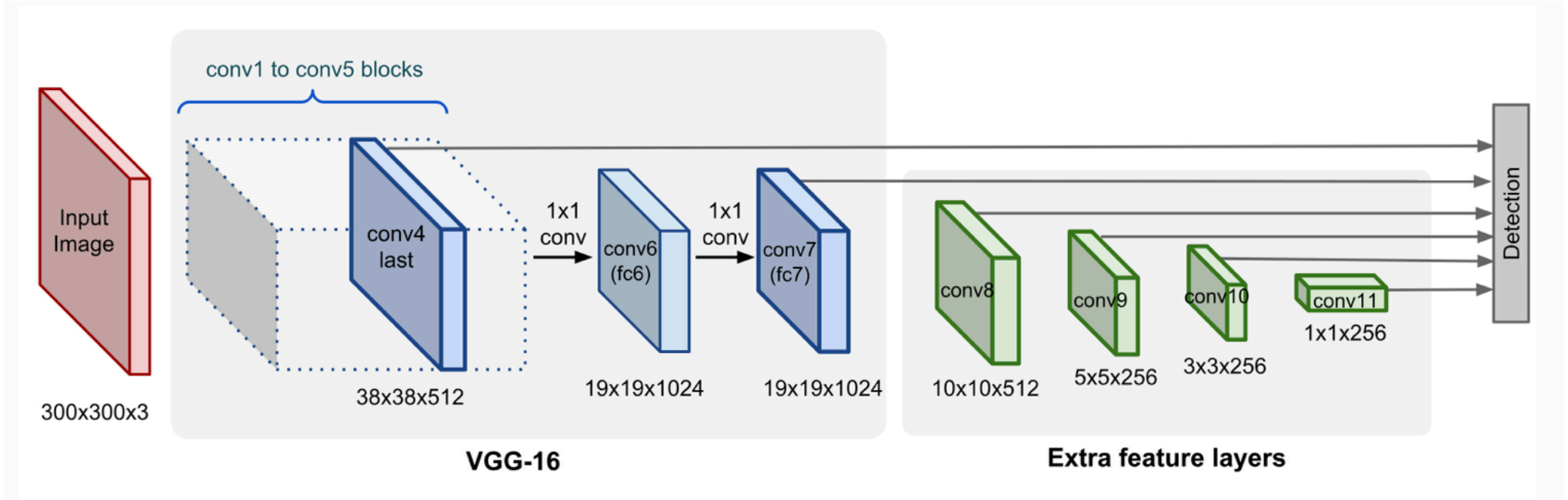
YOLO v1, v2, v3 ⋯ v4, v5

YOLO v4, v5..



SSD

Version..



YOLO v1, v2, v3https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html#yolo-you-only-look-once

https://curt-park.github.io/2017-03-26/yolo/

https://pjreddie.com/darknet/yolo/

Multi-feature map

level index: $\ell = 1, \ldots, L$

scale of boxes: $s_\ell = s_{\min} + \frac{s_{\max} - s_{\min}}{L-1}(\ell - 1)$

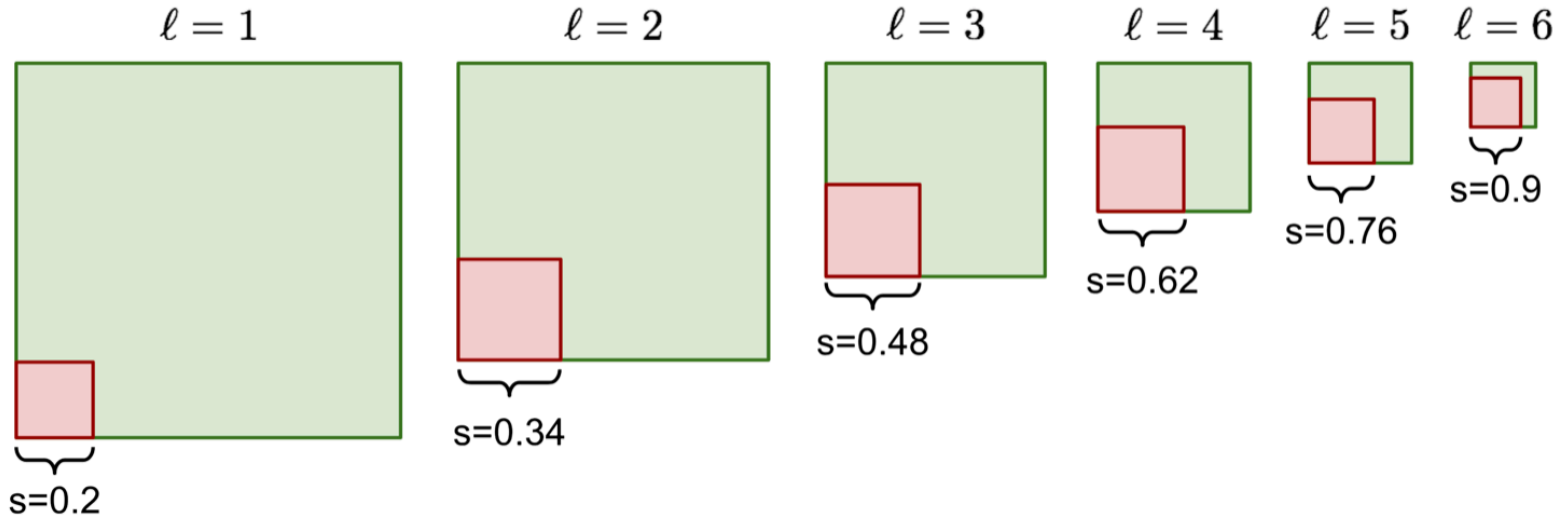
aspect ratio: $r \in \{1, 2, 3, 1/2, 1/3\}$

additional scale: $s'_\ell = \sqrt{s_\ell s_{\ell+1}}$ when $r = 1$thus, 6 boxes in total.

width: $w^r_\ell = s_\ell \sqrt{r}$

height: $h^r_\ell = s_\ell / \sqrt{r}$

center location: $(x^i_\ell, y^j_\ell) = (\frac{i+0.5}{m}, \frac{j+0.5}{n})$



Reference

Reference

https://m.blog.naver.com/jws2218/222077974368

https://m.blog.naver.com/laonple/220918802749 **라온피플**

https://curt-park.github.io/2017-03-17/faster-rcnn/ **Faster RCNN**

https://m.blog.naver.com/jws2218/222077974368 **NMS**

https://curt-park.github.io/2017-03-26/yolo/ **Yolo**

https://towardsdatascience.com/evolution-of-yolo-yolo-version-1-afb8af302bd2

https://towardsdatascience.com/yolo-v4-or-yolo-v5-or-pp-yolo-dad8e40f7109 **Yolo version**