

Object Detection

콥스랩
류태선

Contents

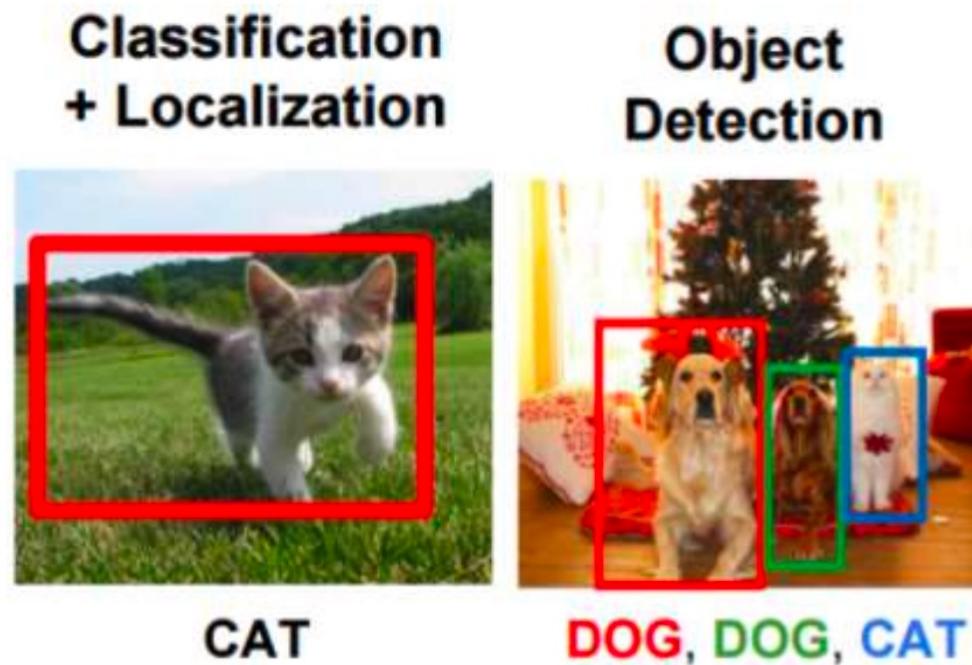
<Object Detection Models>

- R-CNN (2013)
- Fast R-CNN (2015)
- Faster R-CNN (2016)

Introduction

■ Object Detection

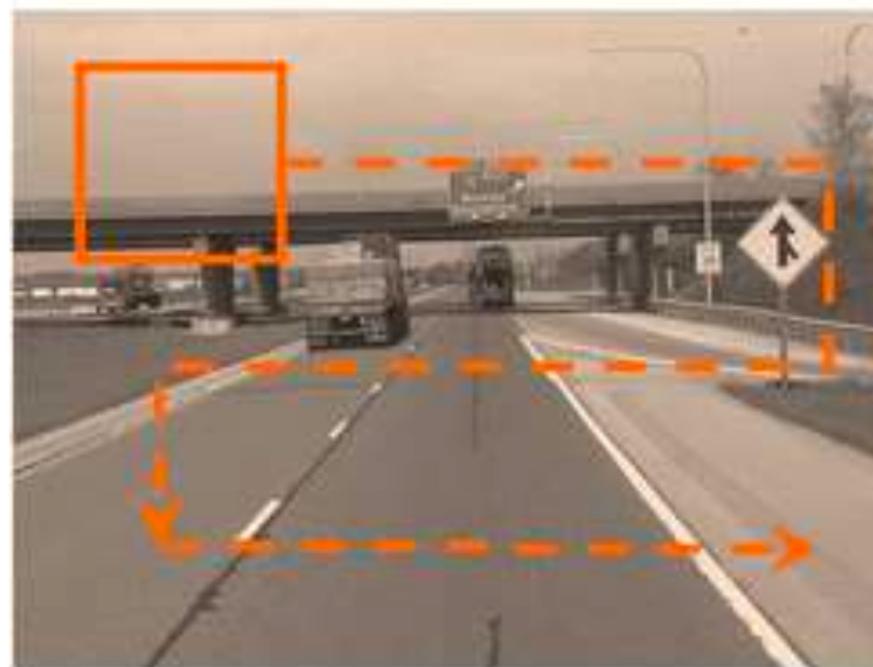
- 입력 이미지 내에 존재하는 모든 카테고리에 대해 classification과 localization을 수행
- Object 0 ~ N개



Naïve Approach

■ Sliding Window Approach

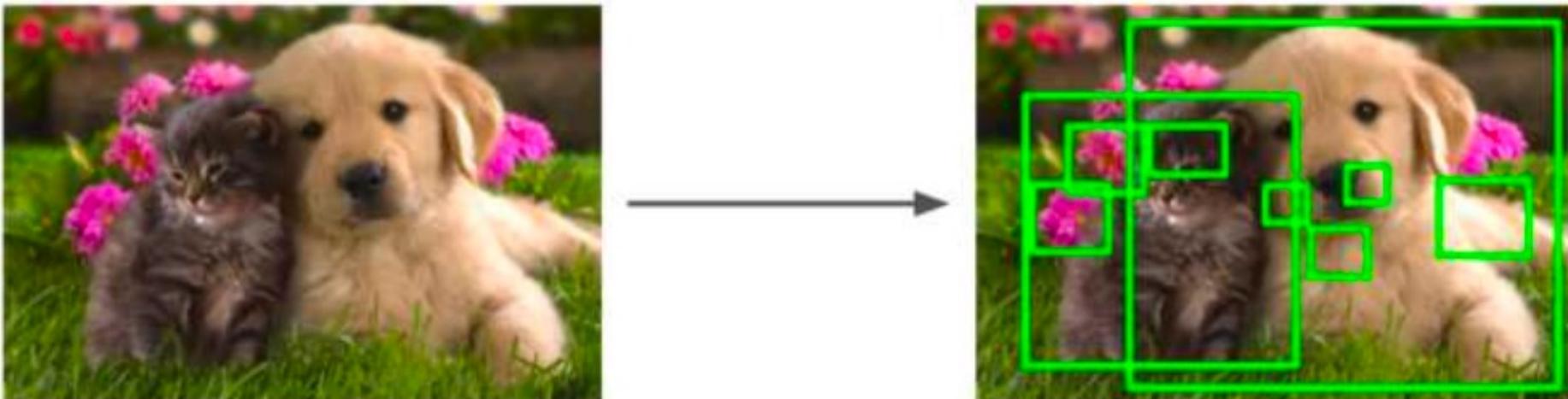
- 모든 크기의 영역 (region with different scale & ratio) 에 대해 sliding window 방식으로 이미지를 모두 탐색하면서 classification을 수행
- 연산량이 많아 비효율적



Region Proposals

■ Region proposal algorithm

- 이미지 상에서 물체가 있을 법한 영역을 찾아내는 알고리즘
- Search space를 줄이기 위한 알고리즘
- Region Proposal Algorithm 예시
 - Sliding Window, Selective Search, Edge Boxes, ...



Selective Search

■ Goal

- 영상은 계층적 구조를 가지므로 적절한 알고리즘을 사용하여 크기에 상관없이 대상을 찾아낸다.
- 컬러, 무늬, 질감, 명암 등 다양한 기준에 따라 segmentation



(a)

(b)



(c)

(d)

Selective Search

■ sub-segmentation

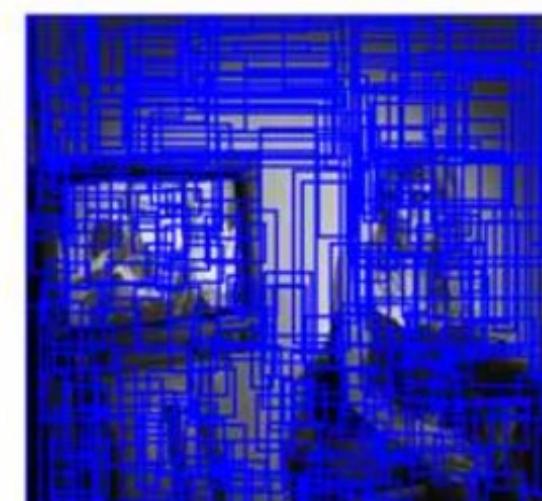
- 각각의 segmentation 객체가 1개의 영역에 할당이 될 수 있도록 많은 초기 영역 (segmentations -> objects) 을 생성한다.



Input Image



Segmentation

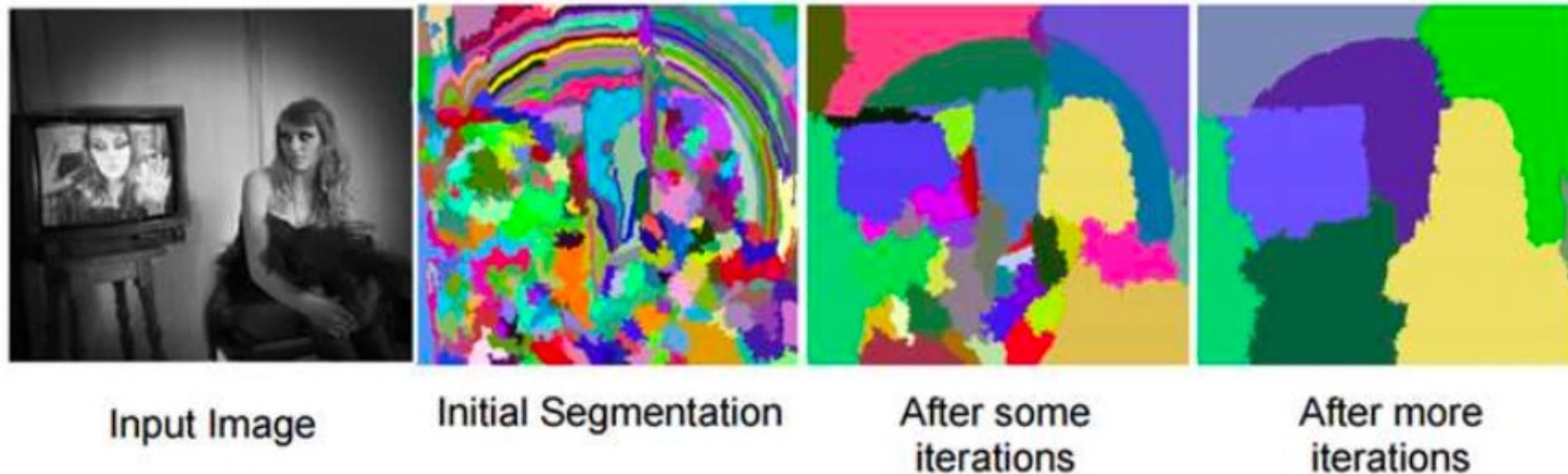


Candidate objects

Selective Search

■ Greedy algorithm

- 여러 영역으로부터 가장 비슷한 영역을 고르고, 유사한 영역과 근접한 Pixel을 점점 Merge, Grouping 해나간다.



- 초기의 작은 영역들이 영역간 유사도에 따라 점점 통합이 되는 것을 확인할 수 있다.

Selective Search

■ Region of Interest (ROI)

- 통합된 영역들을 바탕으로 후보 영역들을 만들어 낸다.
이 과정을 통합적으로 보여주는 과정은 아래와 같다.



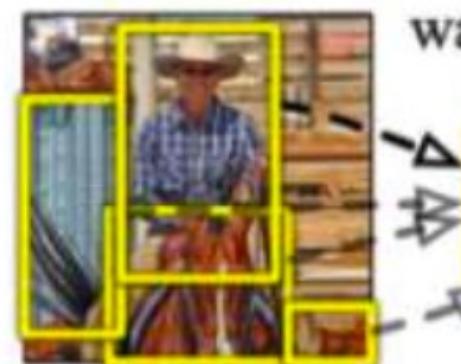
R - CNN

■ Region Proposal + Convolutional Neural Network (CNN)

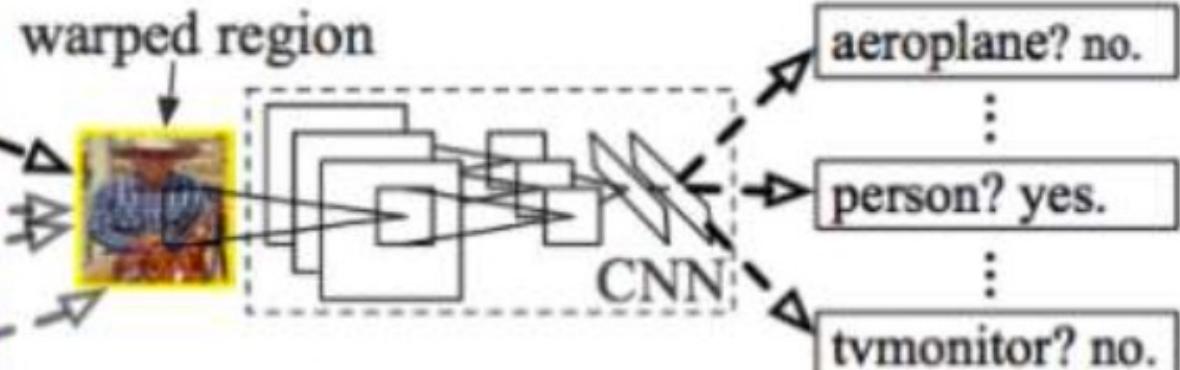
R-CNN: *Regions with CNN features*



1. Input image



2. Extract region proposals (~2k)



3. Compute CNN features

4. Classify regions

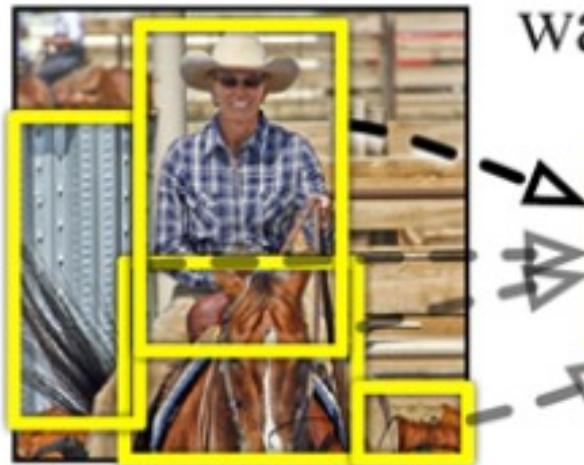
R - CNN

■ Procedure

- 1. Input 이미지에 대해서, 우선 이미지에 대한 후보 영역 (region Proposal)을 생성한다. (약 2000개)



1. Input
image

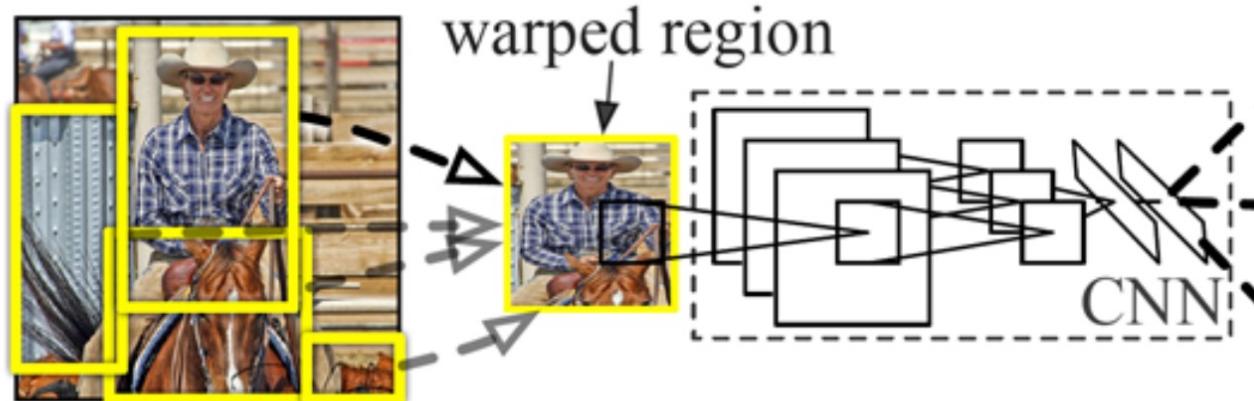


2. Extract region
proposals (~2k)

R - CNN

■ Procedure

- 2. 각 region proposal마다 고정된 크기로 wraping/crop하여 CNN 인풋으로 사용한다. 여기서 CNN은 이미 ImageNet을 활용한 pre-trained된 네트워크를 사용한다.



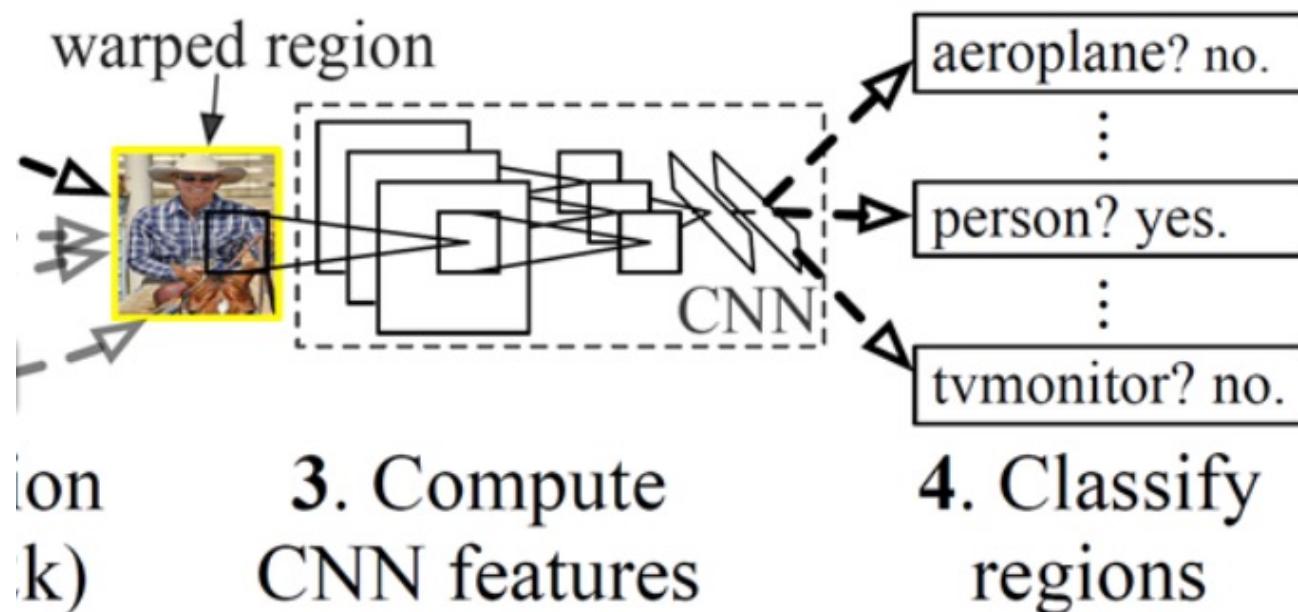
2. Extract region
proposals (~2k)

3. Compute
CNN features

R - CNN

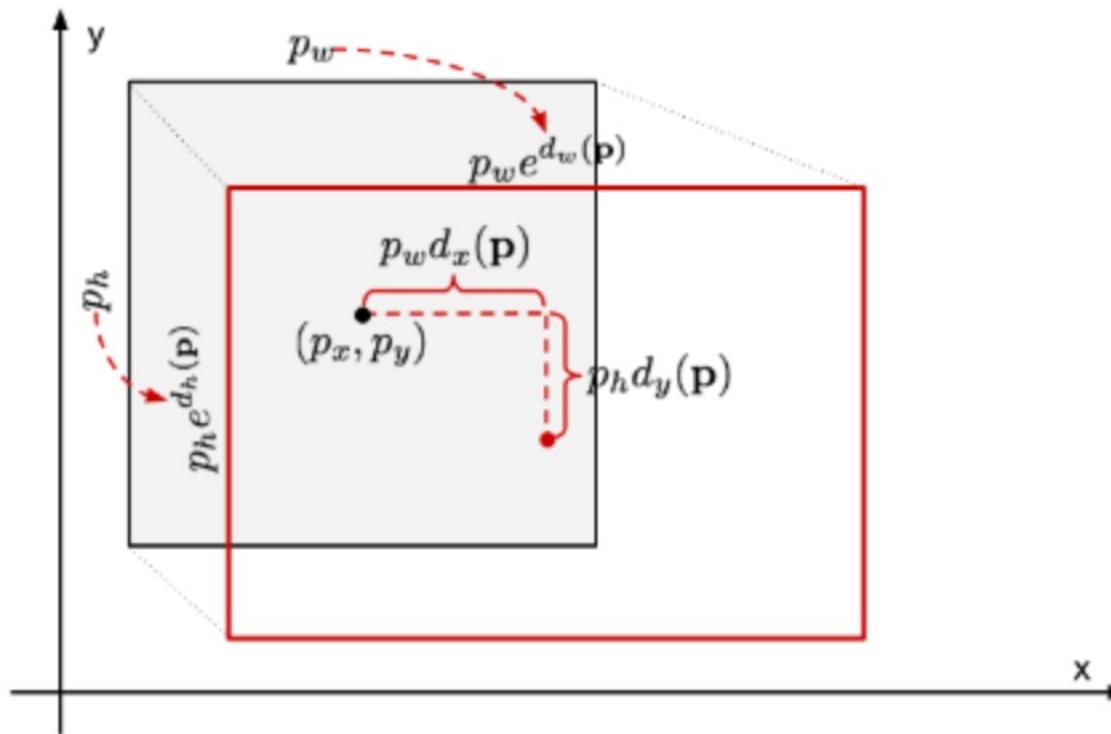
■ Procedure

- 3. CNN을 통해 나온 feature map을 활용하여 SVM을 통한 Object Classification, regressor를 통한 bounding box regression을 진행한다.



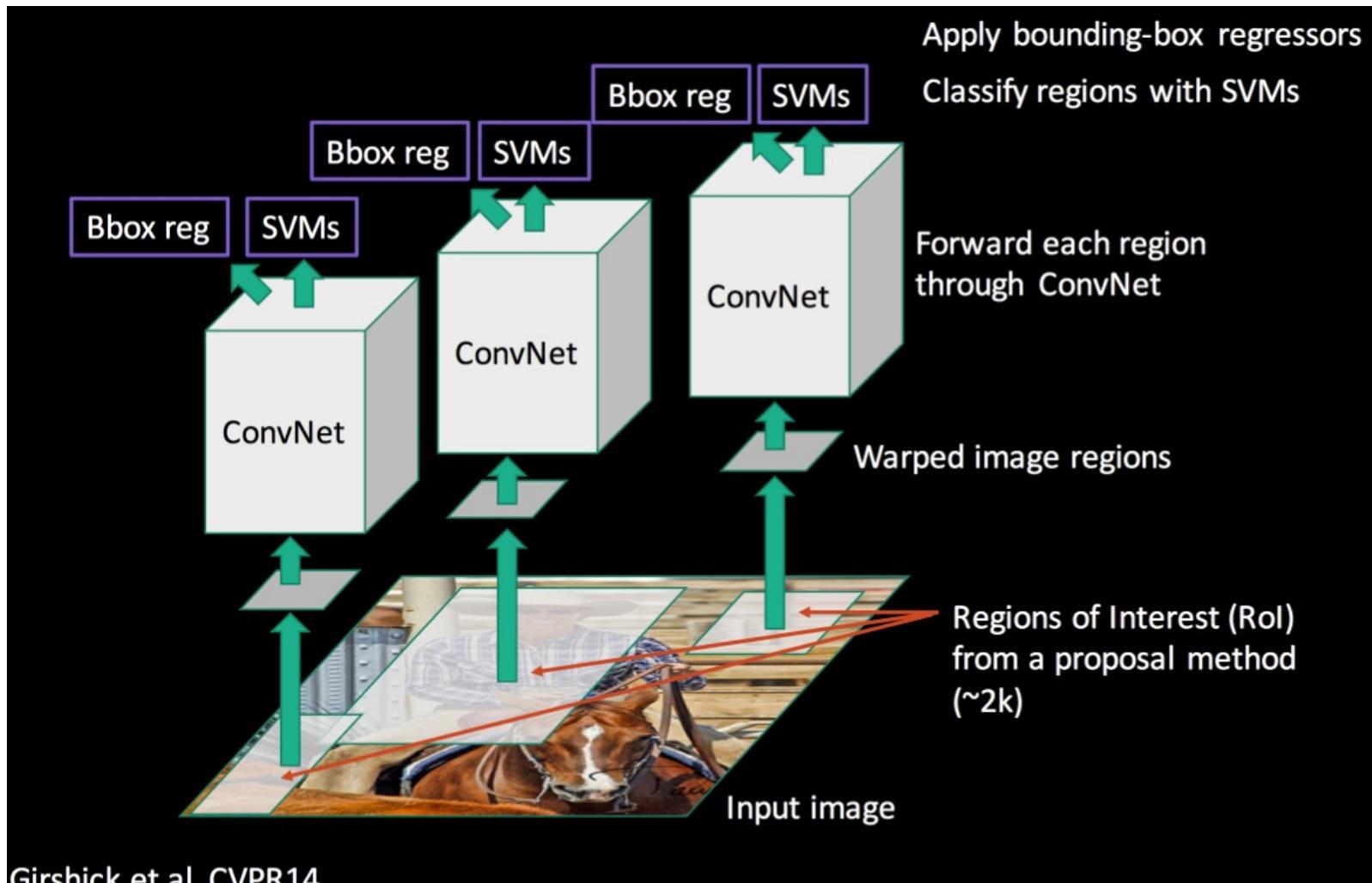
■ Bounding box regression

- Selective Search만 사용했을 경우 Localization 성능이 낮았음.
- 예측한 Bounding box (검은색) 을 실제 정답 Bounding box (붉은색) 으로 보완해 주기 위한 branch.



R - CNN

■ Overall Architecture of R-CNN



R - CNN

■ R-CNN과 타 알고리즘과의 성능 비교

| VOC 2010 test | aero | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv | mAP |
|--------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| DPM v5 [20] [†] | 49.2 | 53.8 | 13.1 | 15.3 | 35.5 | 53.4 | 49.7 | 27.0 | 17.2 | 28.8 | 14.7 | 17.8 | 46.4 | 51.2 | 47.7 | 10.8 | 34.2 | 20.7 | 43.8 | 38.3 | 33.4 |
| UVA [39] | 56.2 | 42.4 | 15.3 | 12.6 | 21.8 | 49.3 | 36.8 | 46.1 | 12.9 | 32.1 | 30.0 | 36.5 | 43.5 | 52.9 | 32.9 | 15.3 | 41.1 | 31.8 | 47.0 | 44.8 | 35.1 |
| Regionlets [41] | 65.0 | 48.9 | 25.9 | 24.6 | 24.5 | 56.1 | 54.5 | 51.2 | 17.0 | 28.9 | 30.2 | 35.8 | 40.2 | 55.7 | 43.5 | 14.3 | 43.9 | 32.6 | 54.0 | 45.9 | 39.7 |
| SegDPM [18] [†] | 61.4 | 53.4 | 25.6 | 25.2 | 35.5 | 51.7 | 50.6 | 50.8 | 19.3 | 33.8 | 26.8 | 40.4 | 48.3 | 54.4 | 47.1 | 14.8 | 38.7 | 35.0 | 52.8 | 43.1 | 40.4 |
| R-CNN | 67.1 | 64.1 | 46.7 | 32.0 | 30.5 | 56.4 | 57.2 | 65.9 | 27.0 | 47.3 | 40.9 | 66.6 | 57.8 | 65.9 | 53.6 | 26.7 | 56.5 | 38.1 | 52.8 | 50.2 | 50.2 |
| R-CNN BB | 71.8 | 65.8 | 53.0 | 36.8 | 35.9 | 59.7 | 60.0 | 69.9 | 27.9 | 50.6 | 41.4 | 70.0 | 62.0 | 69.0 | 58.1 | 29.5 | 59.4 | 39.3 | 61.2 | 52.4 | 53.7 |

Table 1: Detection average precision (%) on VOC 2010 test. R-CNN is most directly comparable to UVA and Regionlets since all methods use selective search region proposals. Bounding-box regression (BB) is described in Section C. At publication time, SegDPM was the top-performer on the PASCAL VOC leaderboard. [†]DPM and SegDPM use context rescoring not used by the other methods.

R - CNN

■ R-CNN의 문제점

■ Test 속도가 느림

- 모든 region proposal에 대한 CNN feature를 추출
- GPU(K40)에서 13s / image

■ SVM (Classification) 과 Bounding box regressor의 학습이 분리

- CNN 학습 과정 후, SVM과 bounding box regressor의 학습이 나중에 진행됨 (post-hoc)

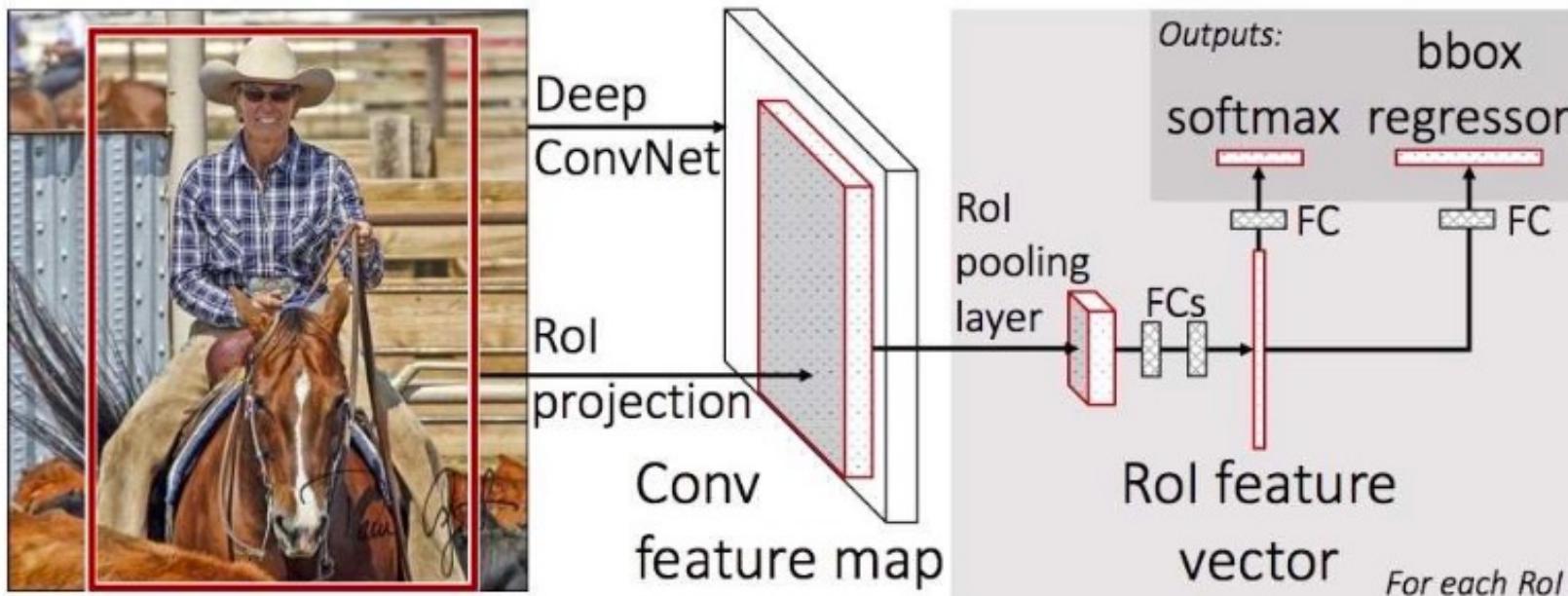
■ 학습 과정이 복잡하고 오래걸림

- 여러 단계를 거치는 training pipeline
- GPU(K40)에서 84시간 (VOC2007, 5000 images)

Fast R - CNN

■ 특징: R-CNN의 속도 개선

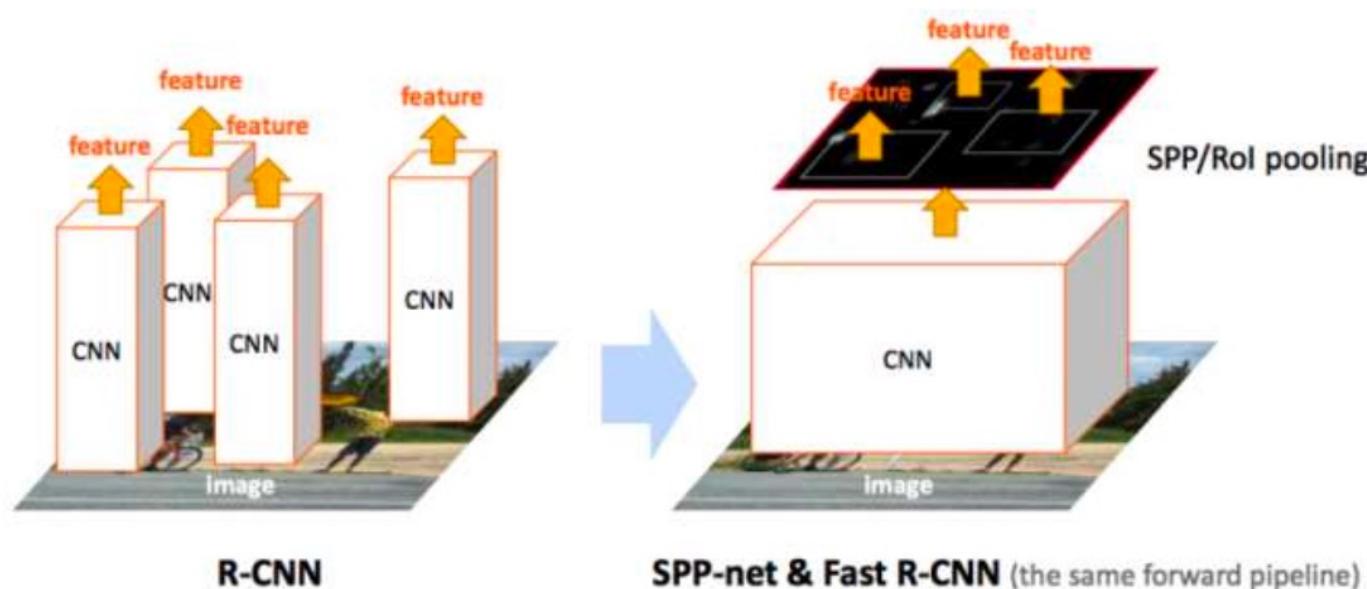
- 같은 image의 Region proposal들이 convolution layer를 공유
- ROI Pooling 도입
- 전체 network가 End-to-end로 한번에 학습
- ~160x faster than R-CNN



Fast R - CNN

■ R-CNN 과 (Region Proposal) Feature 추출 방법 비교

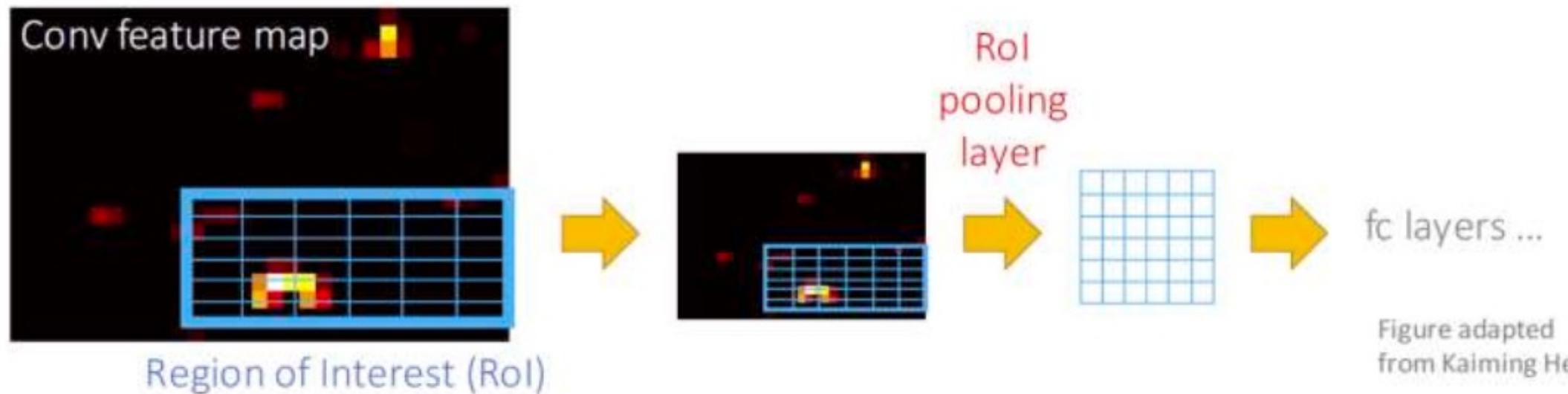
- R-CNN은 image level에서 region proposal을 각각 crop하여 CNN 연산 진행
=> 총 2000개의 region proposal에 CNN 연산
- Fast R-CNN은 전체 이미지를 CNN에 통과시킨 feature map 상에서 Region proposal을 찾음
=> (input 이미지에서 Selective search를 통해 찾은 RoI feature map에 적용 시킨 것
=> CNN 연산을 2000번에서 **1번**으로 줄임



Fast R - CNN

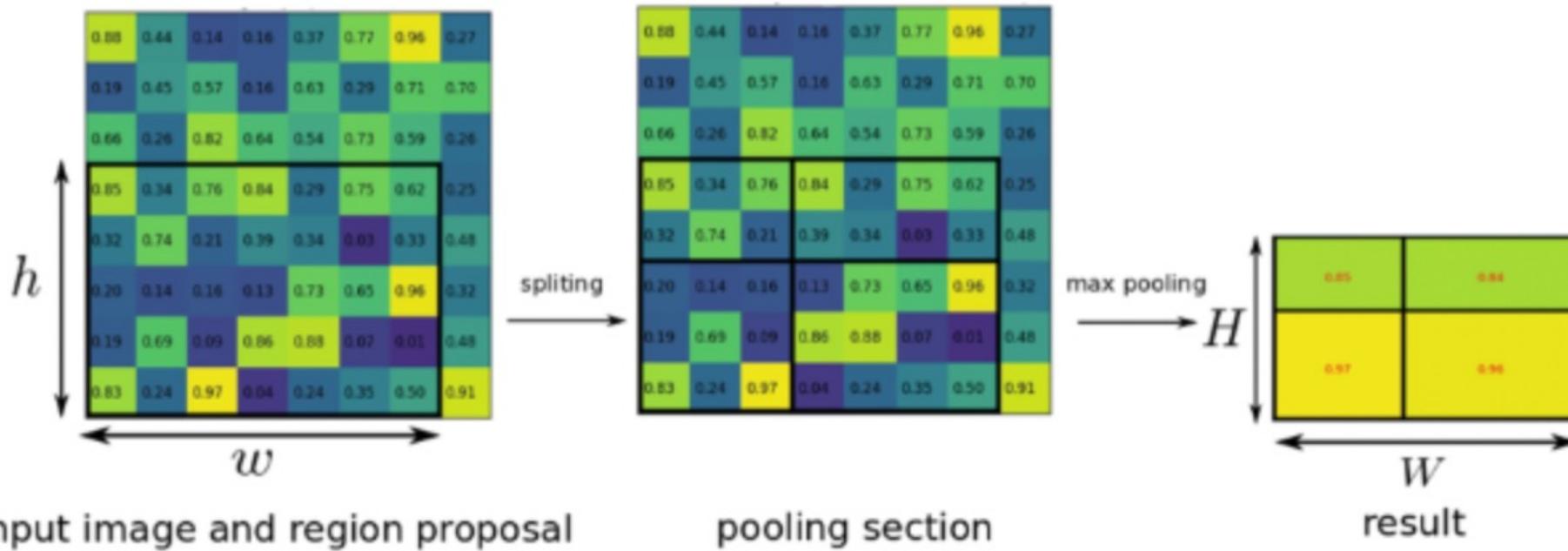
■ ROI pooling (1)

- Selective Search를 통해 찾은 ROI 영역 전체 image를 convolution 해서 나오는 feature map에서 ROI 영역만 pooling 하여 fc layer에 넣는다.



Fast R - CNN

■ ROI pooling (2)

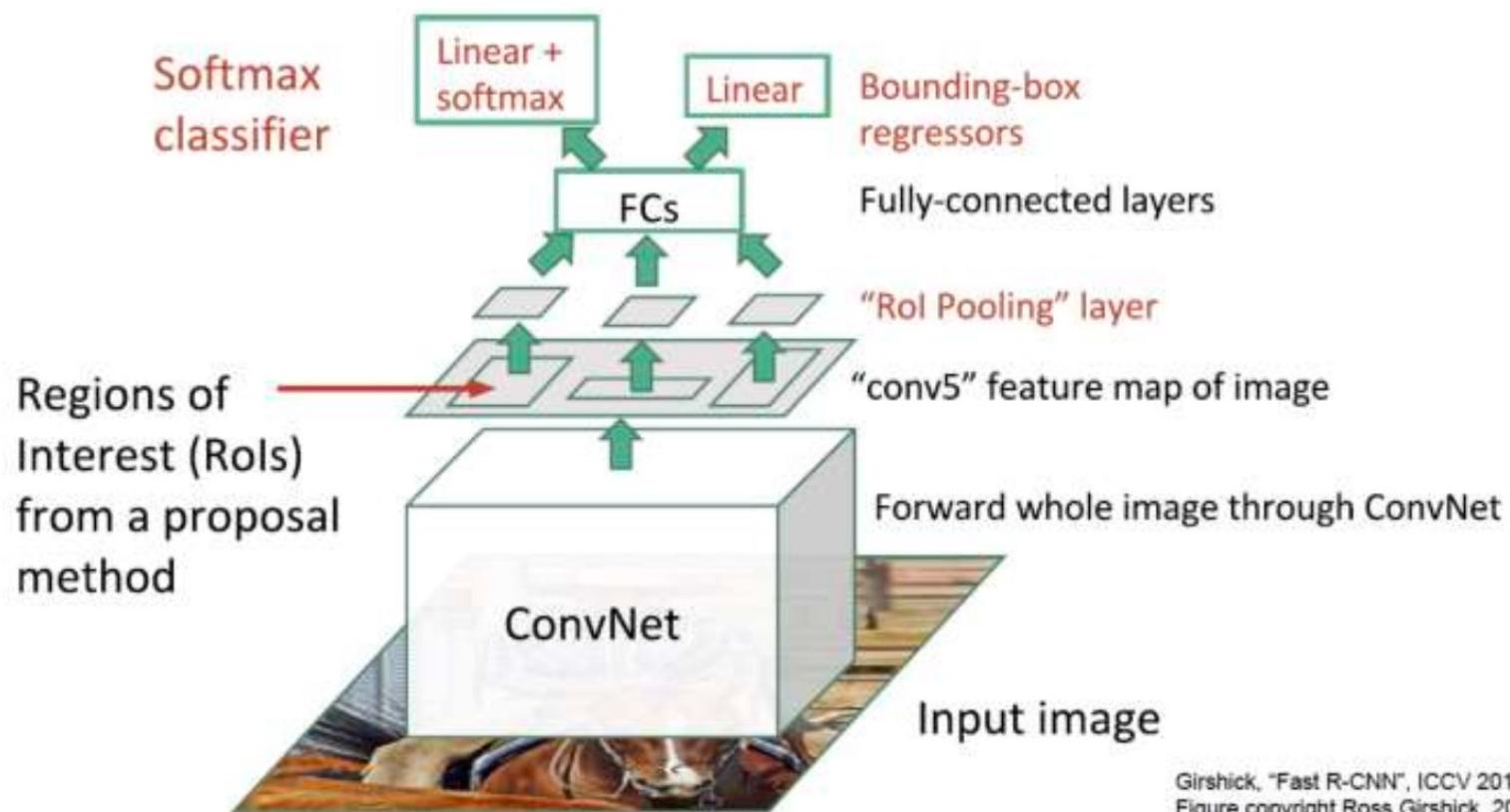


1. Selective Search를 통해서 얻은 feature map 상에서의 ROI (h by w rectangle)
2. ROI($h \times w$)를 $H \times W$ 의 고정된 작은 윈도우 사이즈로 나누고, 나눠진 영역에서 max-pooling을 적용
3. 결과값으로 항상 $H \times W$ 크기의 feature map이 생성됨.

⇒ 이런 pooling을 하는 이유는, 다 제각각의 크기인 ROI 영역을 동일한 크기로 맞춰 fc (fully connect) layer로 넘기기 위함

Fast R - CNN

■ Overall Architecture of Fast R- CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015;

Fast R - CNN

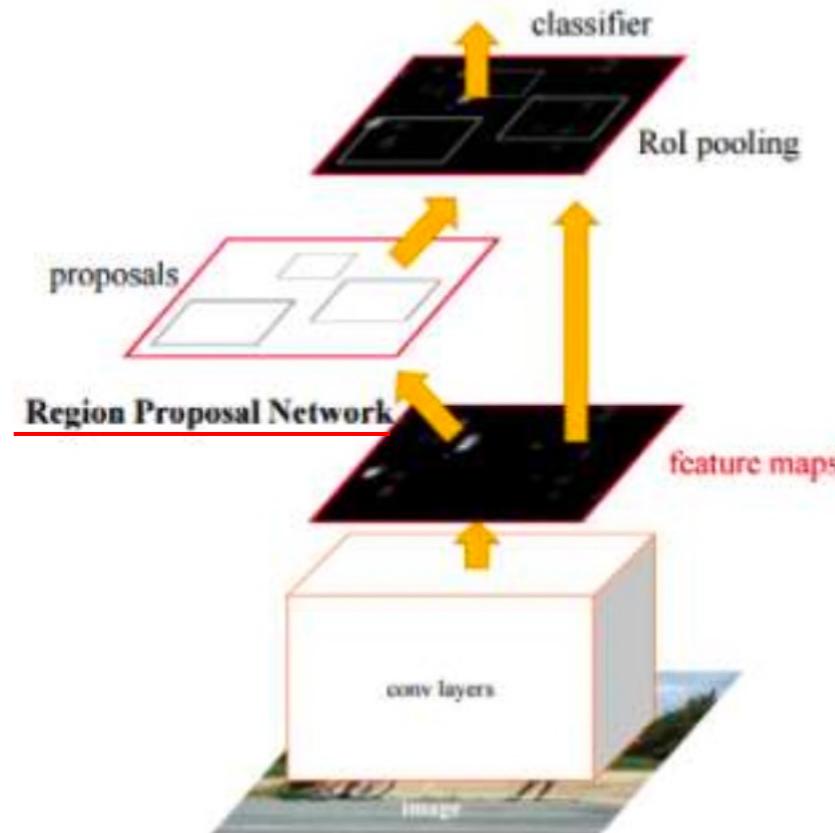
■ Fast R-CNN의 문제점

- Region proposal 을 selective search 로 수행하기에 Region proposal 연산이 느리다는 단점이 있음.
 - Region proposal (Selective Search)가 전체 성능의 bottleneck 이 된다.
⇒ Selective search의 성능에 따라 최종 classification / localization 성능에 크게 영향을 미침
- => 느린 이유중 하나는 GPU가 아니라 CPU로 계산하기 때문
=> Region proposal algorithm을 GPU 연산을 사용할 수 있게 만들어보자!

Faster R - CNN

■ RPN (Region Proposal Network) 를 도입

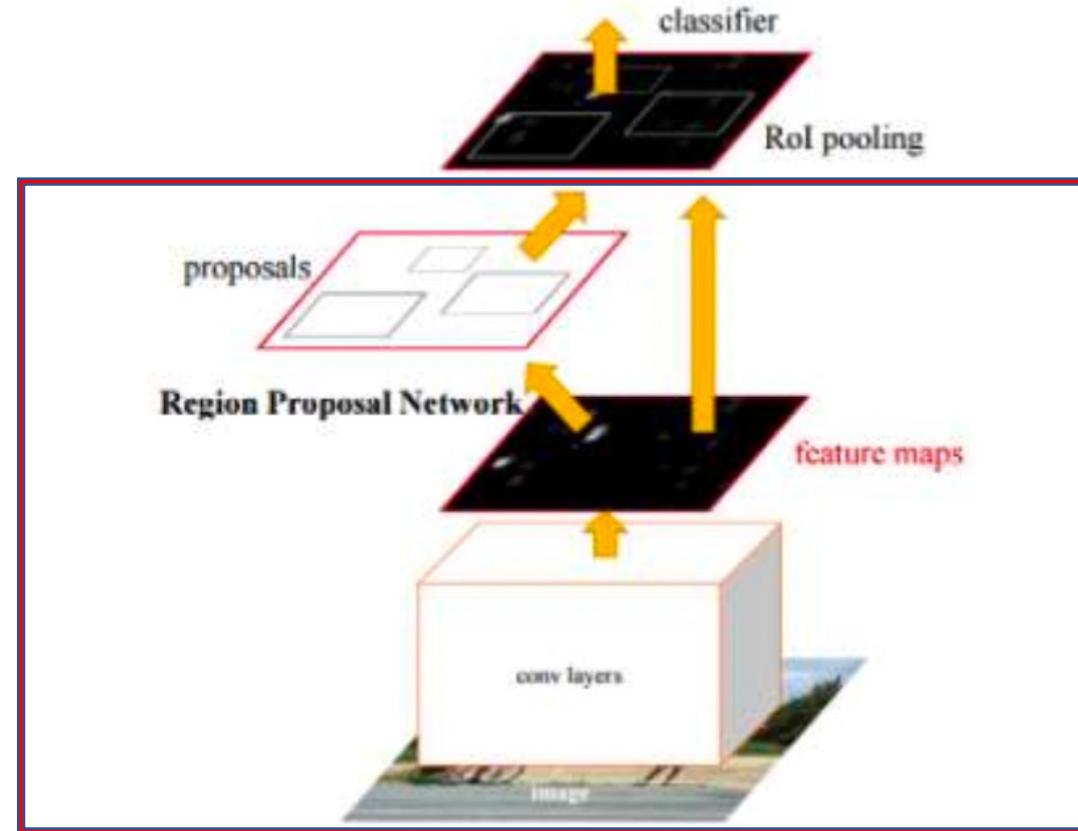
- Fast R-CNN에서 BottleNeck 이었던 Region Proposal 생성알고리즘 (Selective Search)을 CNN 내부에 설계



Faster R - CNN

■ Preview of Faster R-CNN

- 1) 원본 이미지를 pre-trained된 CNN 모델에 입력하여 feature map을 얻음.
- 2) feature map은 RPN에 전달되어 적절한 region proposals을 산출.

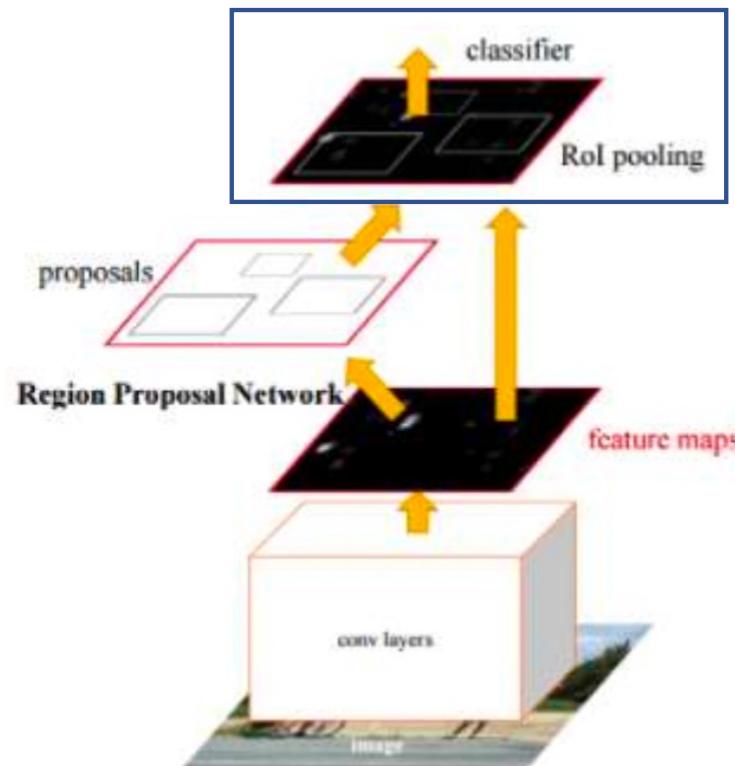


Faster R - CNN

■ Preview of Faster R-CNN

3) region proposals와 1) 과정에서 얻은 feature map을 통해 RoI pooling을 수행하여 고정된 크기의 feature map을 얻음.

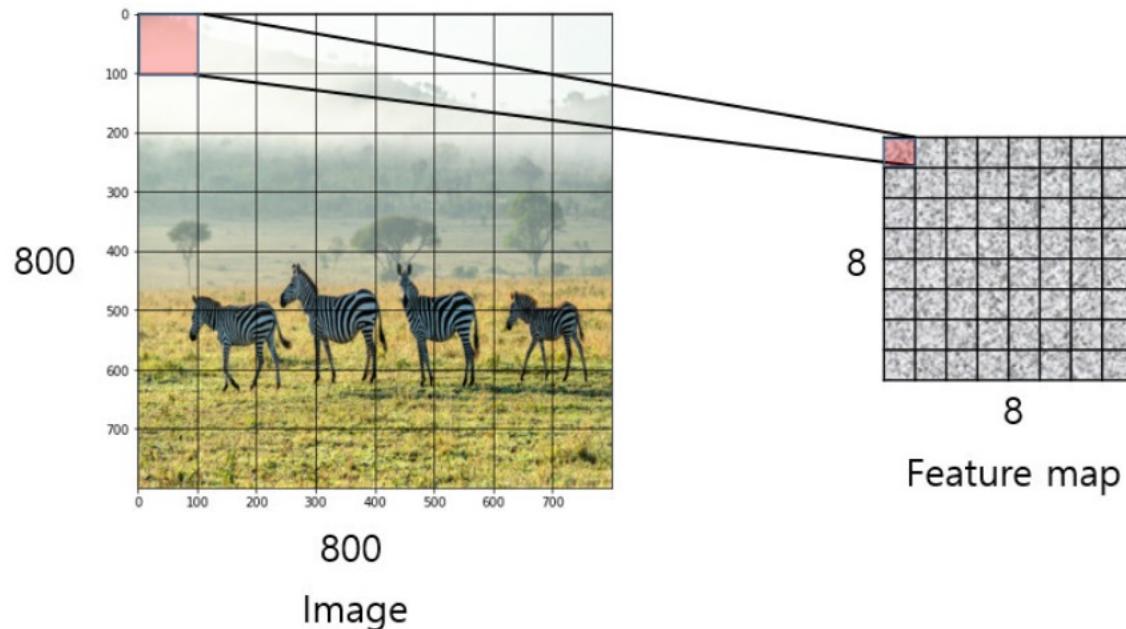
4) Fast R-CNN 모델에 고정된 크기의 feature map을 입력하여 Classification과 Bounding box regression을 수행합니다.



Faster R – CNN

■ Main Idea : Anchor Box

- Selective search 대신에, 원본 이미지를 일정 간격의 grid로 나눔.
- 각 grid cell을 bounding box로 간주하여 feature map에 encode 하는 **Dense Sampling** 방식을 사용
- sub-sampling ratio를 기준으로 원본 이미지를 grid로 나누게 됨.



Ex) 원본이미지 크기가 800x800이고, sub-sampling ratio가 1/100 일때.

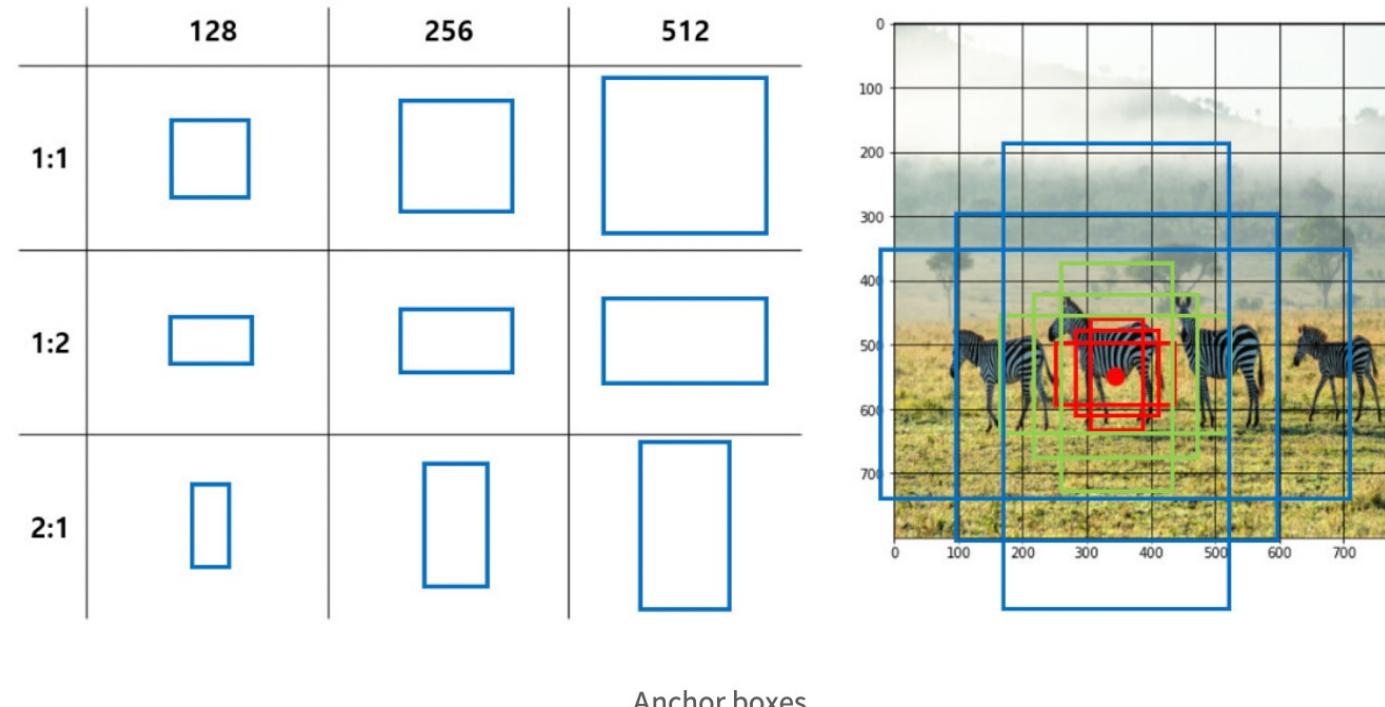
=> CNN을 통과시킨 최종 feature map의 크기는 8x8 (800x1/100)

=> 원본 이미지에서는 8x8개만큼의 bounding box가 생성된다고 볼 수 있습니다.

Faster R – CNN

■ Anchor Box

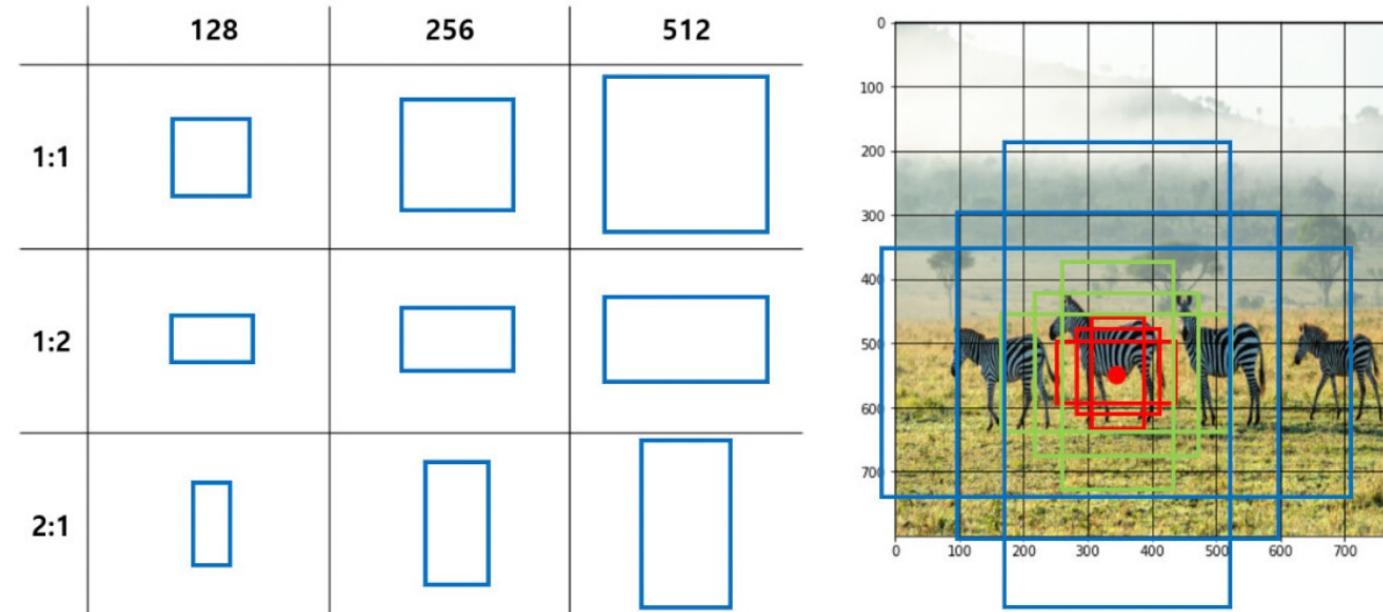
- 하지만, 고정된 크기의 bounding box를 사용한다면 다양한 크기와 모양의 Object들을 잡아내는 데에 무리가 있음.
- 지정한 위치에, 사전에 정의한 서로 다른 크기(scale)와 가로 세로비(aspect ratio)를 가지는 bounding box인 **Anchor box**를 생성하여 다양한 크기의 객체를 포착하는 방법을 제시



Faster R – CNN

■ Anchor Box

- 논문에서는 3가지 scale(크기) “128,256,512”, 3가지 ratio(비율) “2:1, 1:1, 1:2” 사용
- # scale x # ratio = **9 predefined Anchors** (K=9)

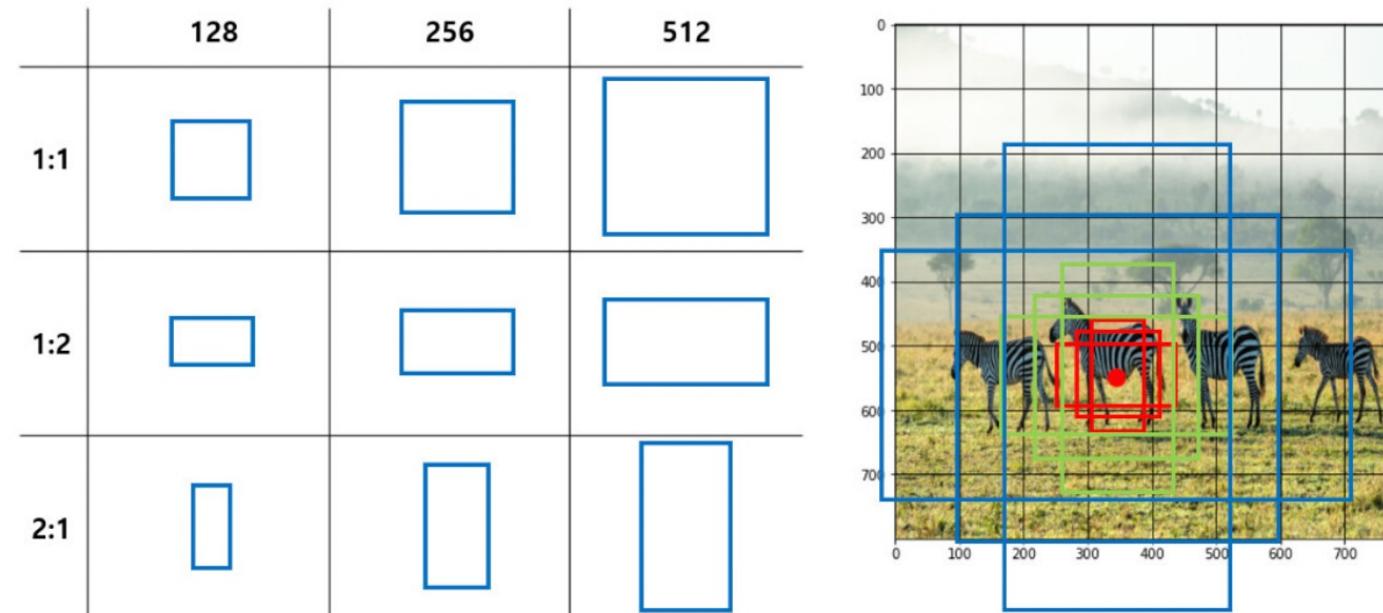


Anchor boxes

Faster R - CNN

■ Anchor Box

- scale : bounding box의 width & height 의 크기
- ratio : bounding box의 width : height 비율 (동일한 scale에서 bounding box의 넓이가 같도록 유지)

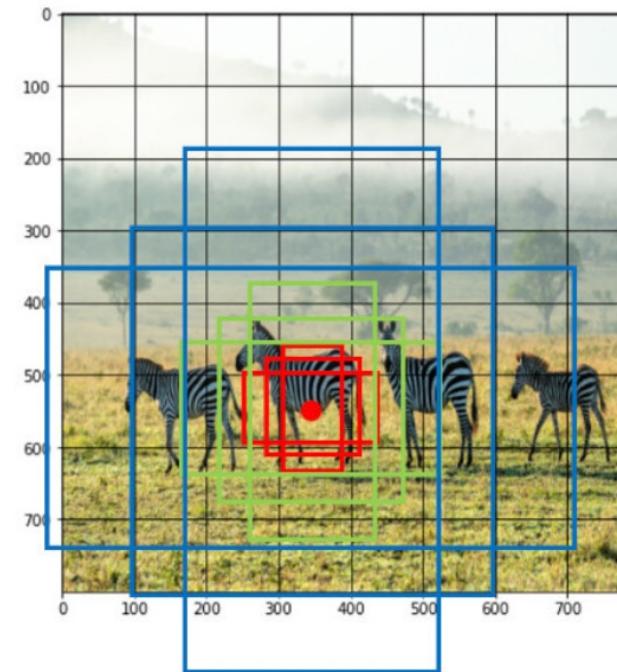
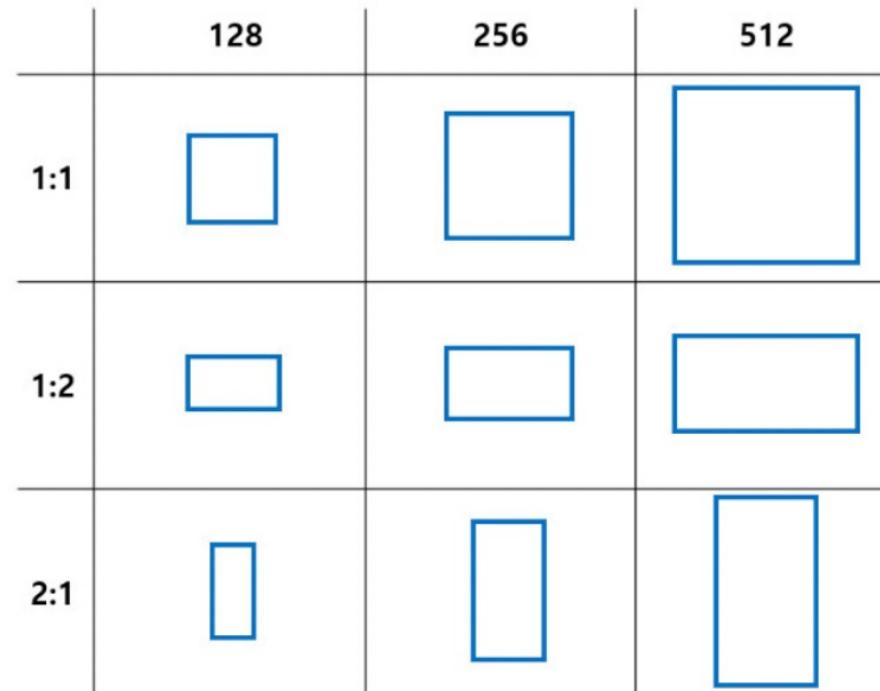


Anchor boxes

Faster R - CNN

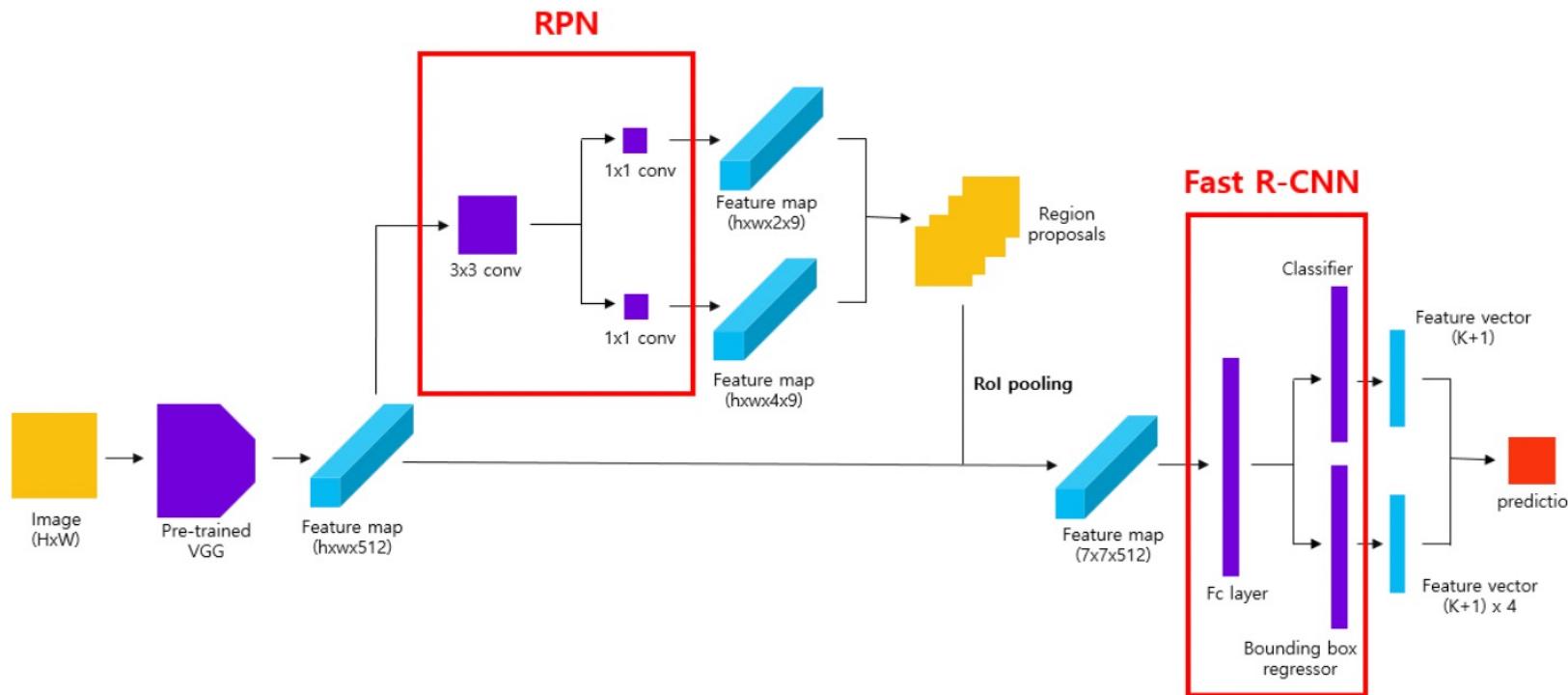
■ Anchor Box

- feature map의 크기가 $W \times H$ 일때
- 총 $W \times H \times K$ 개의 anchor를 가진다. (각 feature의 spatial location마다 K 개의 anchor)



Faster R - CNN

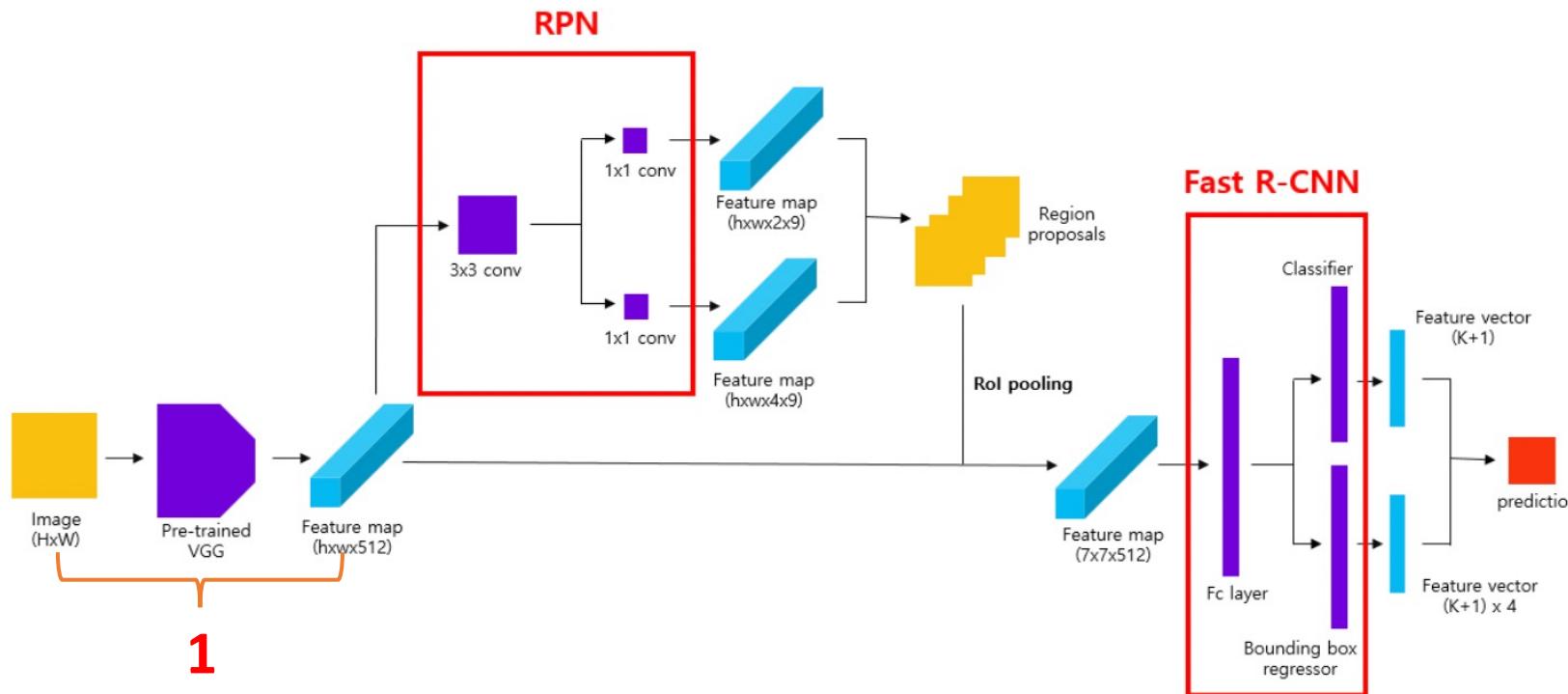
■ Main Idea : RPN (Region Proposal Network)



- RPN은 원본 이미지에서 **region proposals**를 추출하는 네트워크
- 원본 이미지에서 anchor box를 생성하면 수많은 region proposals가 만들어짐.
- RPN은 Anchor box로 부터 생성된 region proposals에 대하여 **class score**를 매기고, **bounding box coefficient**를 출력하는 기능

Faster R - CNN

■ RPN (Region Proposal Network) 동작 예시

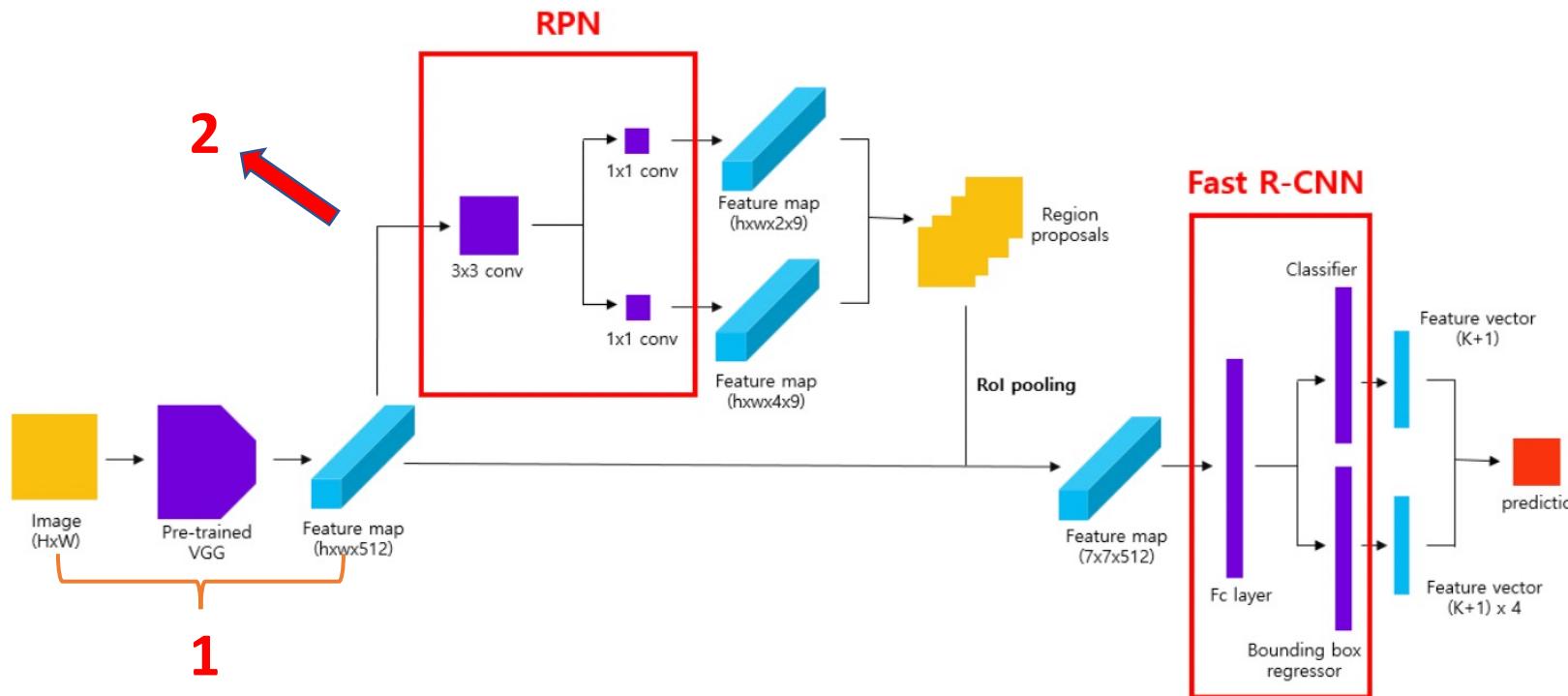


1. 원본 이미지를 pre-trained 된 VGG 모델에 입력하여 feature map을 얻음.

ex) 원본이미지 사이즈 800x800x3, sub-sampling ratio 1/100 일시,
Feature map 사이즈는 8x8x512 (512는 VGG를 통과해서 나온 feature map의 channel 수)

Faster R - CNN

■ RPN (Region Proposal Network) 동작 예시

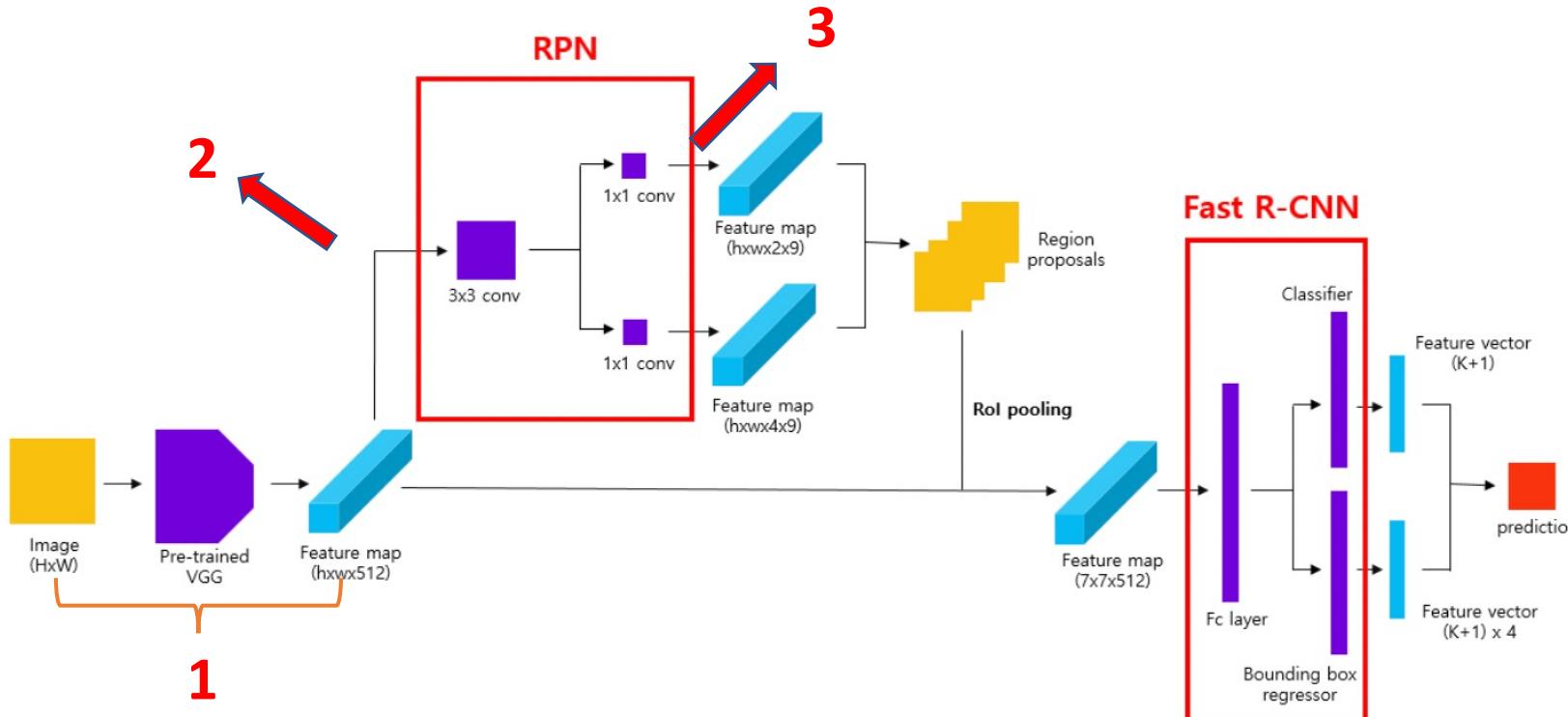


2. 1)에서 얻은 feature map에 대하여 3×3 conv 연산을 적용. 이때 feature map의 크기가 유지될 수 있도록 padding 또한 적용. (intermediate Layer 생성)

ex) $8 \times 8 \times 512$ feature map에 대하여 3×3 연산을 적용하여 $8 \times 8 \times 512$ 개의 feature map이 출력됨.

Faster R - CNN

■ RPN (Region Proposal Network) 동작 예시

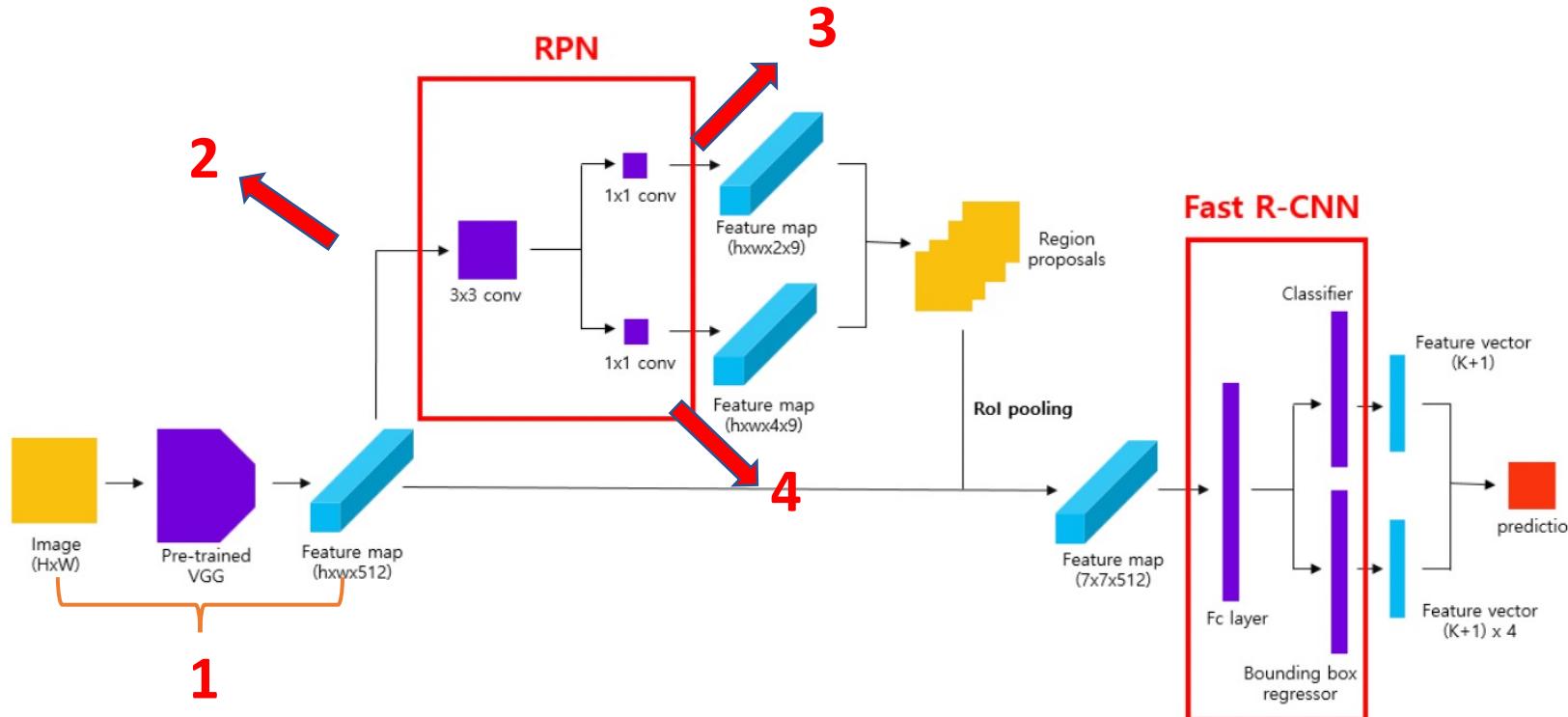


3. class score를 매기기 위해서 feature map에 대하여 **1x1 conv** 연산을 적용하여 출력하는 feature map의 channel 수가 **2x9**가 되도록 만듬. => 9 (# of anchor box) * 2 (background or object)

RPN에서는 region proposal이 어떤 class에 해당하는지까지 구체적인 분류를 하지 않고 객체가 포함되어 있는지 여부만을 확인.

Faster R - CNN

■ RPN (Region Proposal Network) 동작 예시

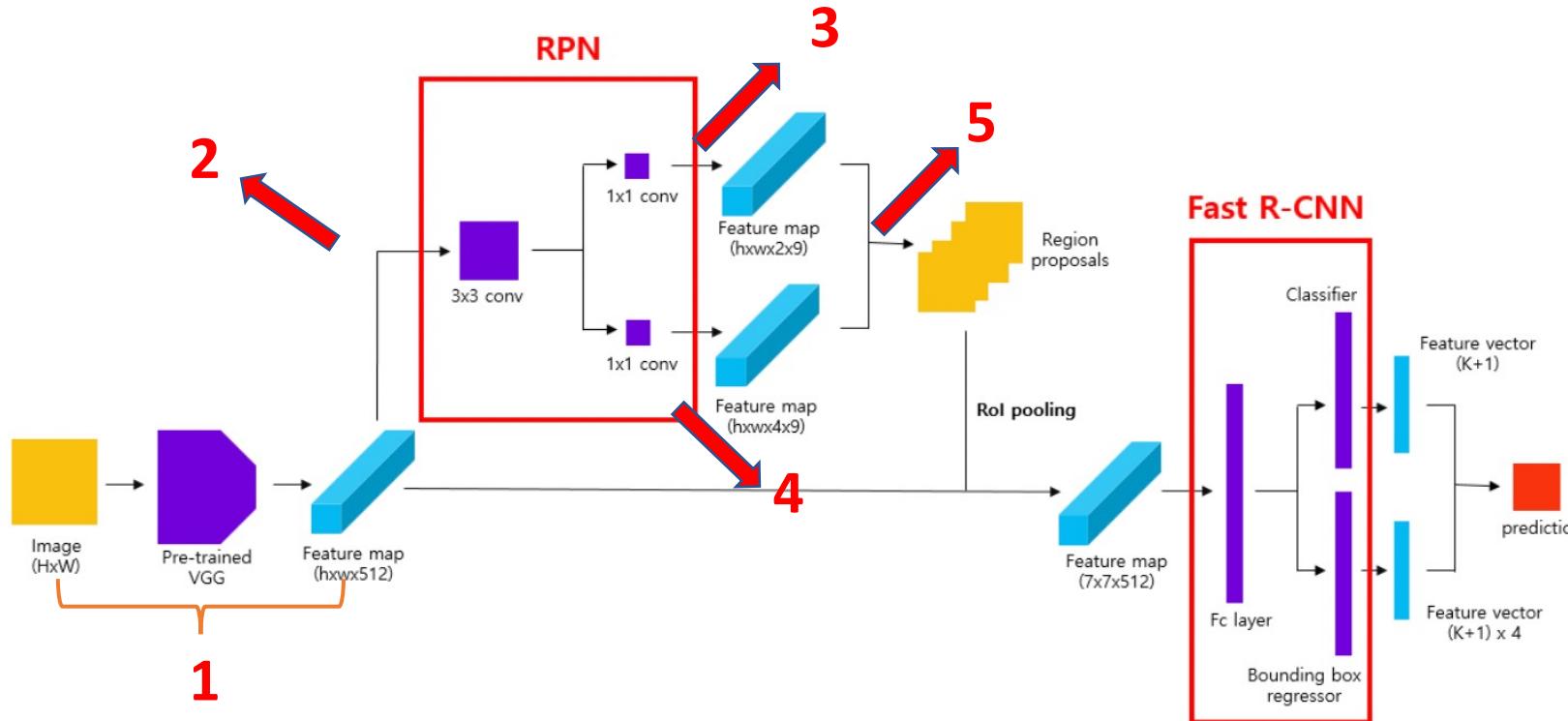


4. bounding box regressor를 얻기 위해 feature map에 대하여 **1x1 conv** 연산을 적용.

이 때 출력하는 feature map의 channel 수가 **4(bounding box regressor) x 9(anchor box 9개)**가 되도록 설정. 이는 Bounding Box 의 localization을 좀 더 정교히 해주는 역할을 함.

Faster R - CNN

■ RPN (Region Proposal Network) 동작 예시



5. 3)에서 얻은 feature를 통해, Classification을 통해서 얻은 물체일 확률 값들을 정렬한 다음, 높은 순으로 N개의 앵커만 추려냄.

N 개의 앵커들에 각각 4)에서 얻은 feature를 바탕으로 Bounding box regression을 적용. 그 다음 Non-Maximum-Suppression을 적용하여 RoI을 구해줌.

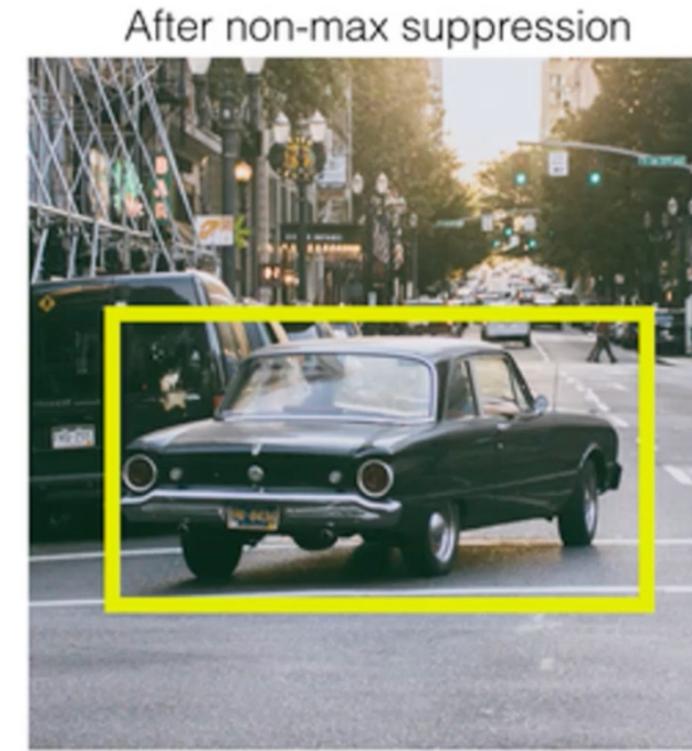
Faster R - CNN

■ Non-Maximum Suppression (NMS)

- 동일한 Object에 다양한 Bounding box가 생길 때, 가장 스코어가 높은 bbox만 남기고 나머지를 제거하는 알고리즘



Non-Max
Suppression
→



Faster R - CNN

■ Non-Maximum Suppression (NMS)

1. Detected 된 bounding box 별로 Confidence threshold 이하의 bounding box는 제거.
2. 가장 높은 confidence score를 가진 box 순으로 내림차순 정렬하고 아래 로직을 모든 box에 순차적으로 적용
 - => 가장 높은 confidence score를 가진 box와 겹치는 다른 box를 모두 조사하여, IoU 가 특정 threshold 이상인 box를 모두 제거. 즉 IoU가 일정 이상인 bounding box는 동일한 물체를 detect 했다고 판단하고 겹치는 box를 제거해 주는 과정.
3. 남아 있는 box만 선택한다.

예시) <https://www.youtube.com/watch?v=YDkjWEN8jNA> 1:20~

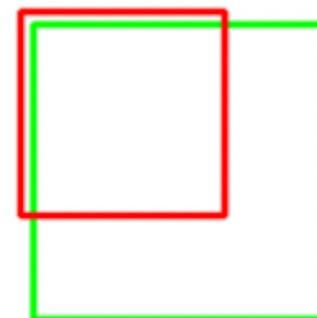
Faster R - CNN

■ IoU (Intersection of Union)

- 두 Bounding box가 얼마나 일치하는지 평가하는 지표.

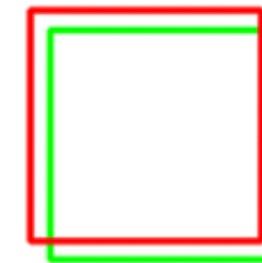
$$\text{IoU} = \frac{\text{Overlapping Region}}{\text{Combined Region}}$$


IoU: 0.4034



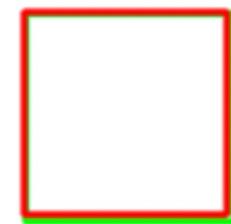
Poor

IoU: 0.7330



Good

IoU: 0.9264

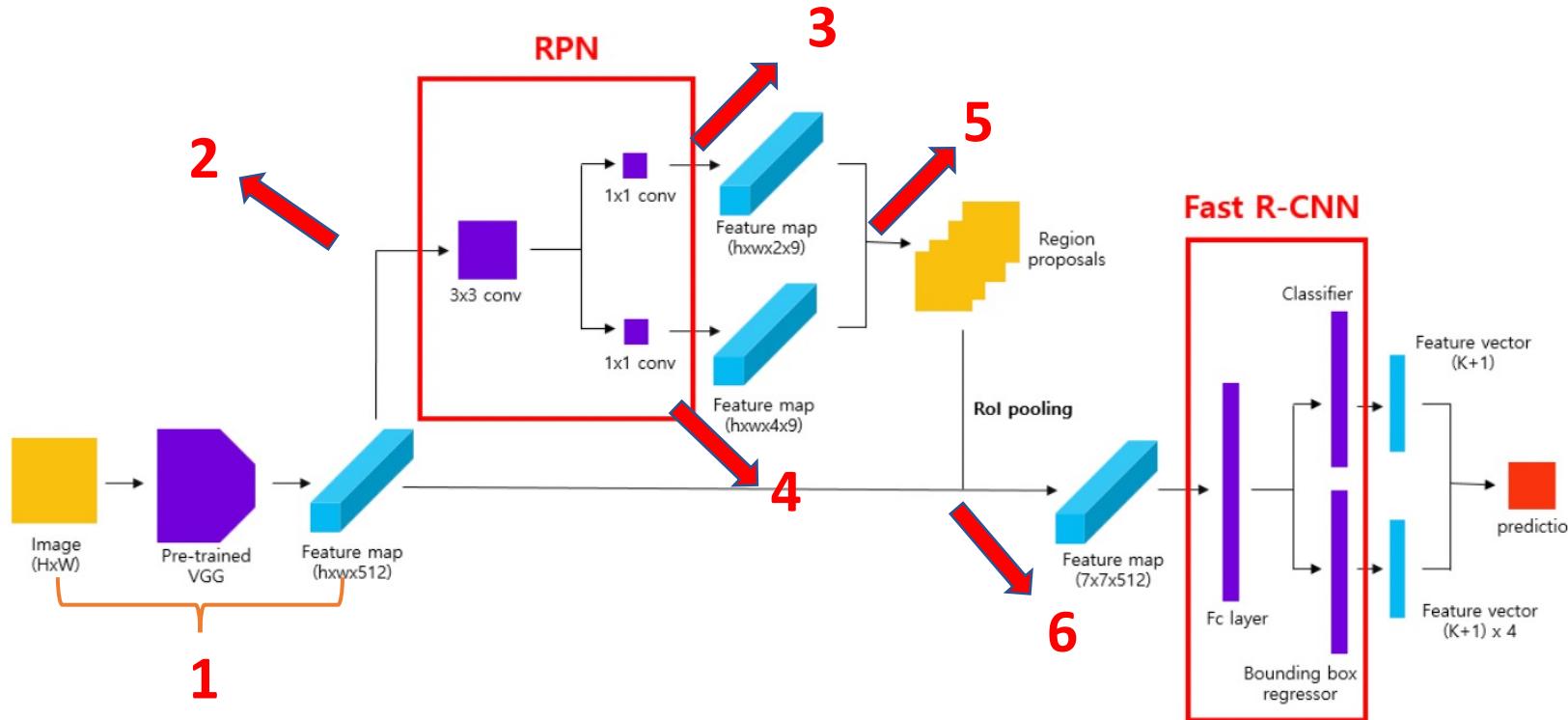


Excellent

IoU = 교집합 영역 넓이 / 합집합 영역 넓이

Faster R - CNN

■ RPN (Region Proposal Network) 동작 예시



6. RPN을 통해 얻은 최종 Region Proposal들을 RoI Pooling을 적용시킴.

이후 RoI Pooling이 적용된 RoI feature들을 Fast-R CNN와 동일한 branch에 전달하게됨.

Faster R - CNN

■ Loss signal of Faster R-CNN

: RPN을 학습시키기 위한 Loss function +
Fast R-CNN 을 학습시키기 위한 Loss function 으로 이뤄짐.

1) RPN Loss Function

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

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

물체가 없으면 bbox는 고려하지 않음.

Faster R - CNN

■ First term of RPN Loss function

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

Predicted Probability of Anchor i

GT Label (물체가 있으면 1, 없으면 0)

물체가 없으면 bbox는 고려하지 않음.

Smooth L1 Loss

predicted coordinates of Anchor i

GT box

- i : mini-batch 내의 anchor의 index
- p_i : anchor i 에 객체가 포함되어 있을 예측 확률
- p_i^* : anchor가 양성일 경우 1, 음성일 경우 0을 나타내는 index parameter
- N_{cls} : mini-batch의 크기(논문에서는 256으로 지정)

$$\textbf{Log loss : } H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

= Binary Cross Entropy. 이진 분류 (Binary Classification) 문제에서 주로 쓰이는 Loss function

Faster R - CNN

■ First term of RPN Loss function

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

Predicted Probability of Anchor i *GT Label* (물체가 있으면 1, 없으면 0) *Smooth L1 Loss*

물체가 없으면 bbox는 고려하지 않음.

predicted coordinates of Anchor i *GT box*

=> RPN Network를 각 Anchor i 안에 Class 가 존재하는지 안하는지를 학습시키기 위해 추가한 term

Faster R - CNN

■ Second term of RPN Loss function

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

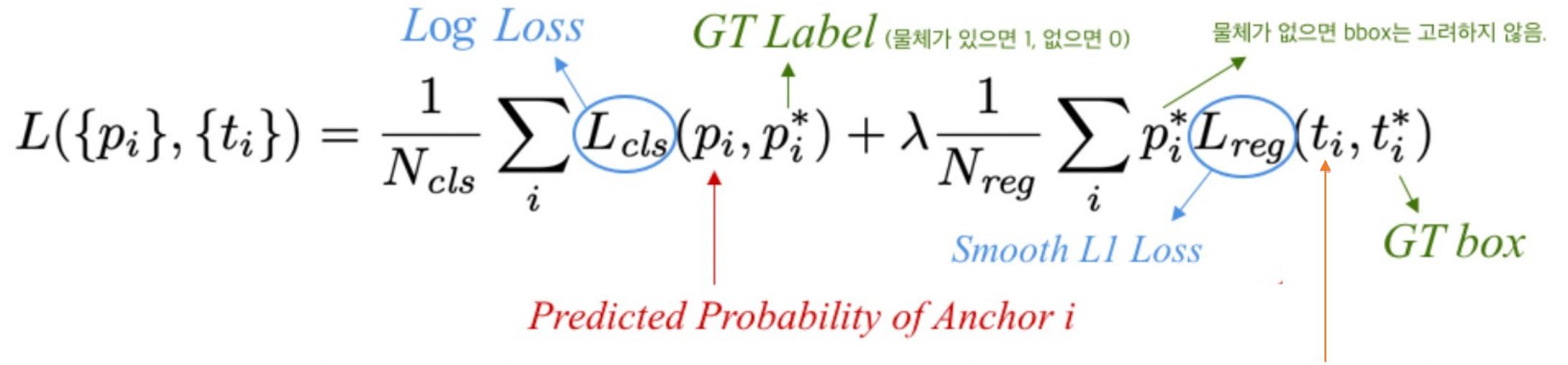
Predicted Probability of Anchor i

GT Label (물체가 있으면 1, 없으면 0)

물체가 없으면 bbox는 고려하지 않음.

Smooth LI Loss

GT box



- i : mini-batch 내의 anchor의 index
- t_i : 예측 bounding box의 파라미터화된 좌표(coefficient)
- t_i^* : ground truth box의 파라미터화된 좌표
- λ : balancing parameter(default=10)
- N_{reg} : anchor 위치의 수
- p_i^* : anchor가 양성일 경우 1, 음성일 경우 0을 나타내는 index parameter

Faster R - CNN

■ Second term of RPN Loss function

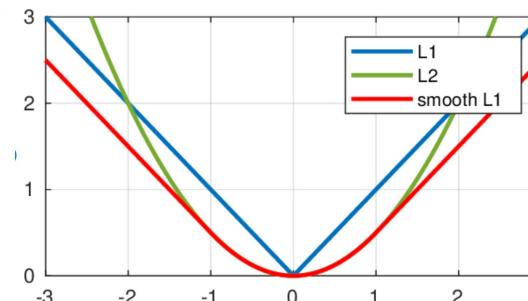
$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

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

Smooth L1 Loss : $\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$

Smooth L1 loss는 L2 loss와 L1 loss의 Combination이라고 보면 된다.

where $x = t_i - t_i^*$



Faster R - CNN

■ Second term of RPN Loss function

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

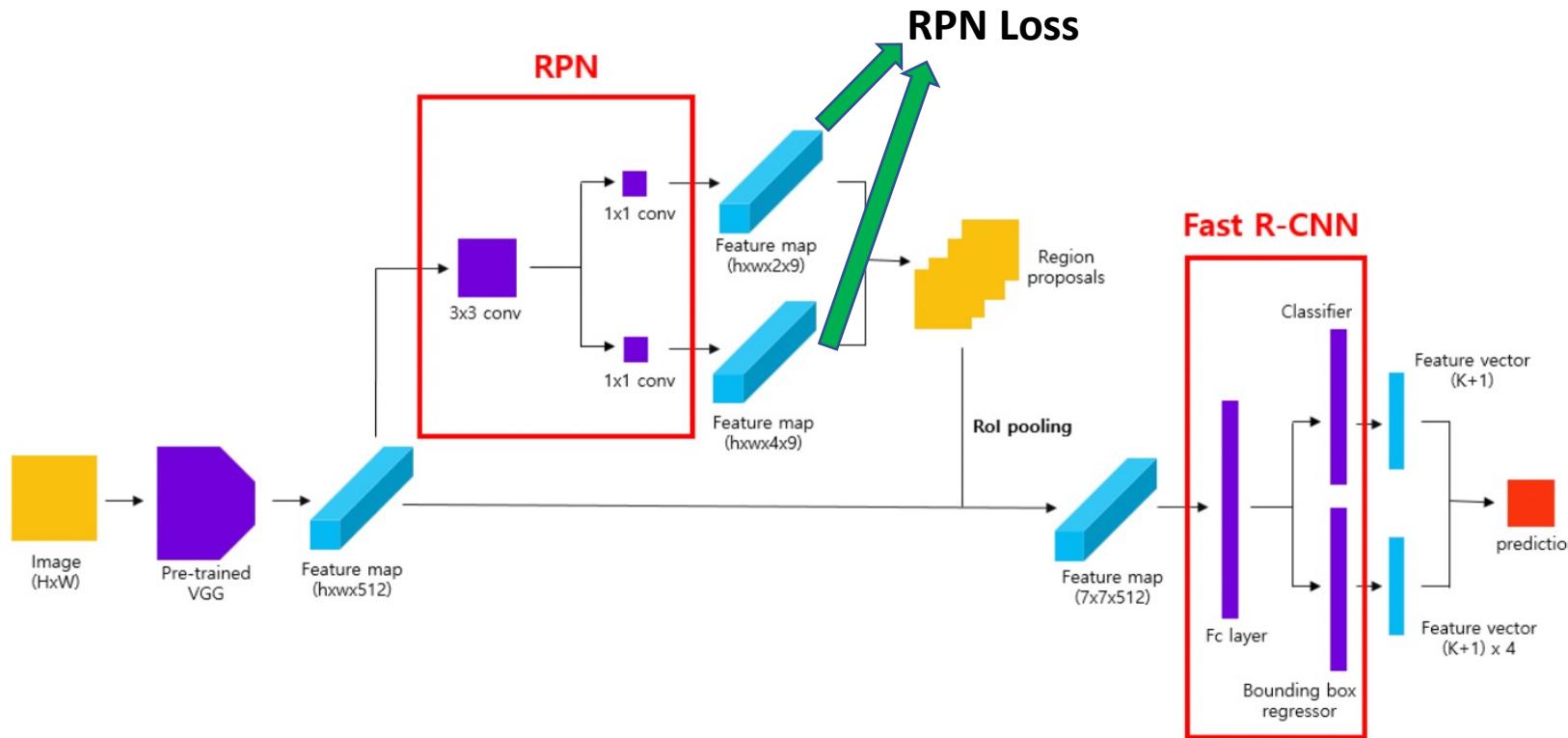
Diagram illustrating the components of the second term of the RPN Loss function:

- Log Loss**: $L_{cls}(p_i, p_i^*)$ is highlighted with a blue circle.
- GT Label**: p_i^* is highlighted with a green arrow pointing to it. A note in Korean: (물체가 있으면 1, 없으면 0)
- Smooth L1 Loss**: $L_{reg}(t_i, t_i^*)$ is highlighted with a blue circle.
- GT box**: t_i^* is highlighted with a green arrow pointing to it.
- Predicted Probability of Anchor i**: p_i is highlighted with a red arrow pointing to it.
- predicted coordinates of Anchor i**: t_i is highlighted with a red arrow pointing to it.
- A note in Korean: 물체가 없으면 bbox는 고려하지 않음.

=> RPN Network의 각 Anchor i 들의 정확한 Bounding box를 예측 (regressor) 할 수 있도록 만드는 역할

Faster R - CNN

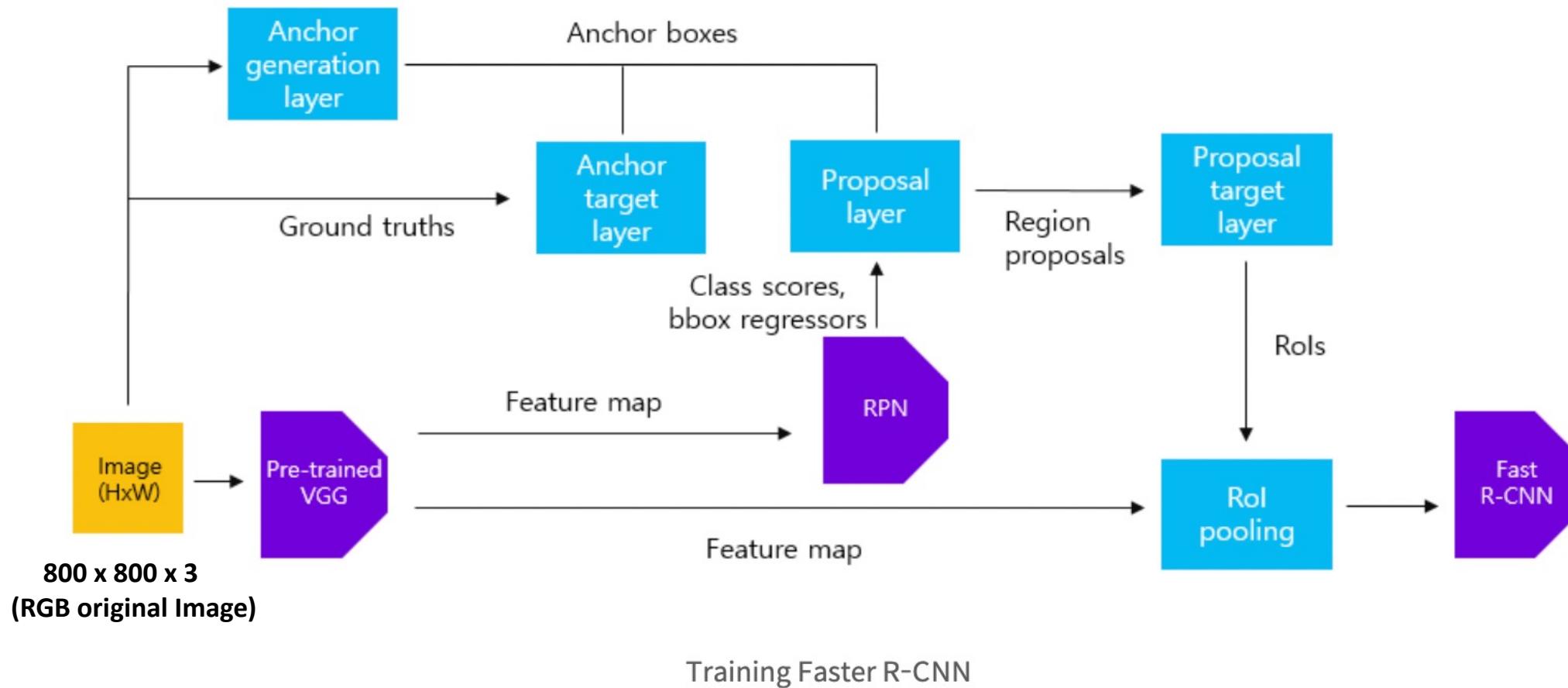
■ RPN Loss function



Faster R - CNN

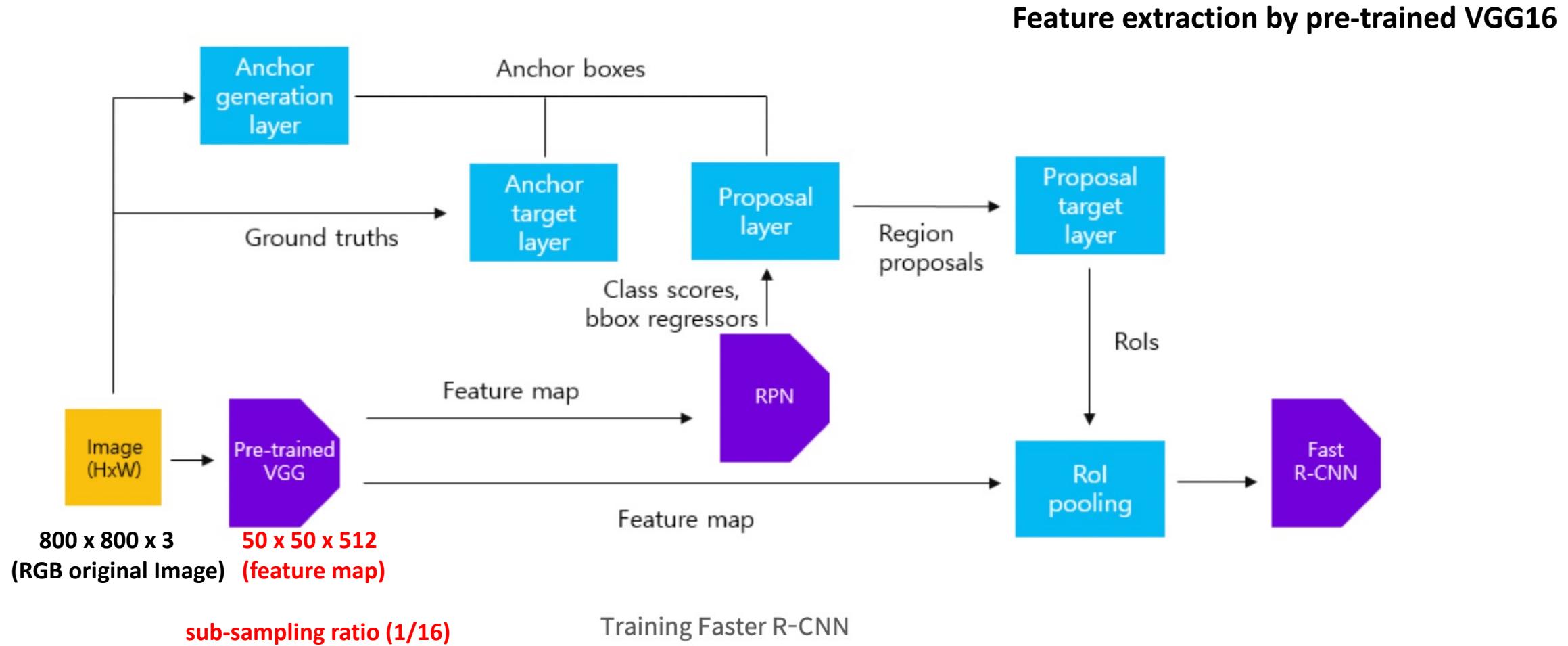
■ Detail of Training Scheme of Faster R-CNN

Origin Image 사이즈 800x800x3



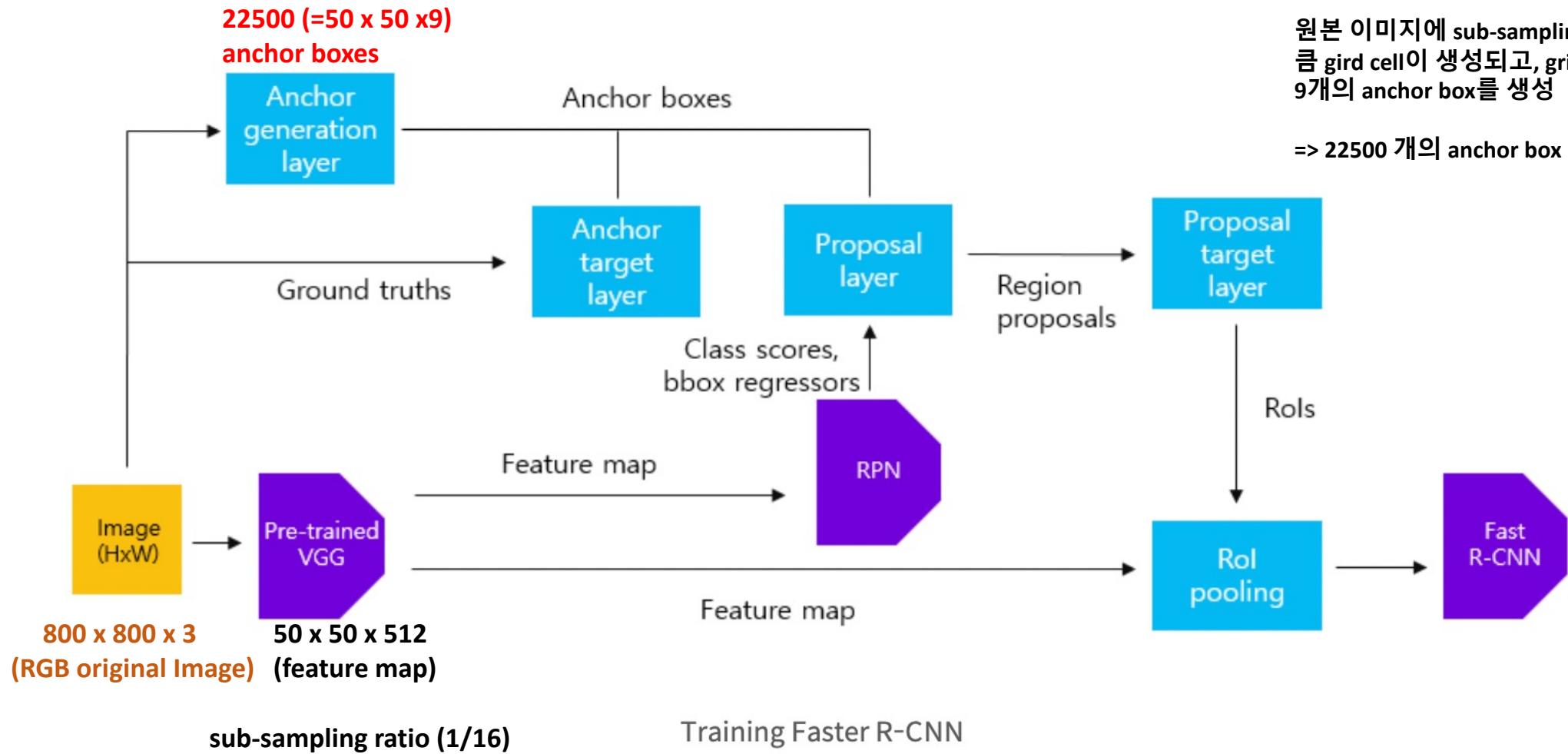
Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN



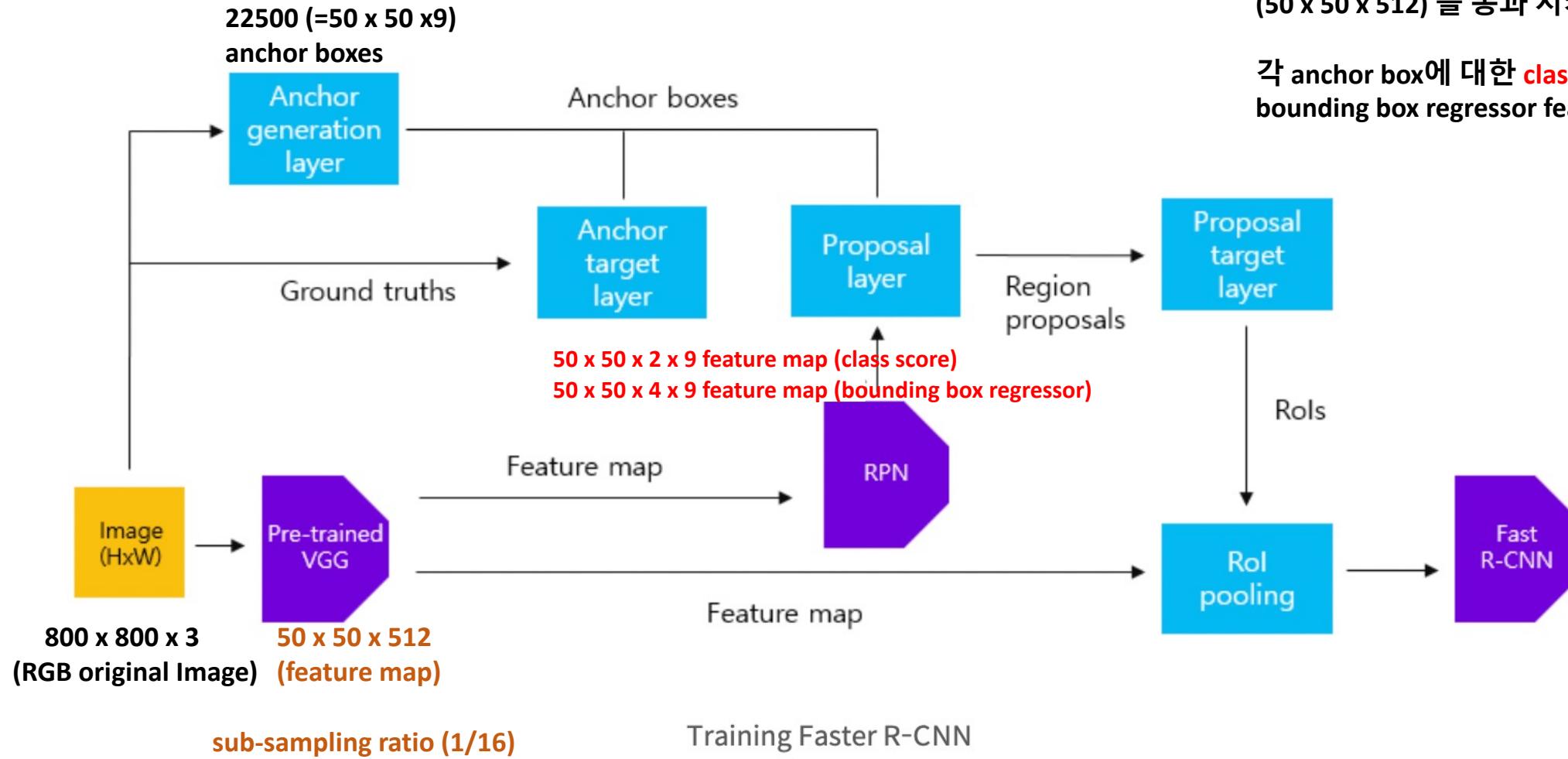
Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN



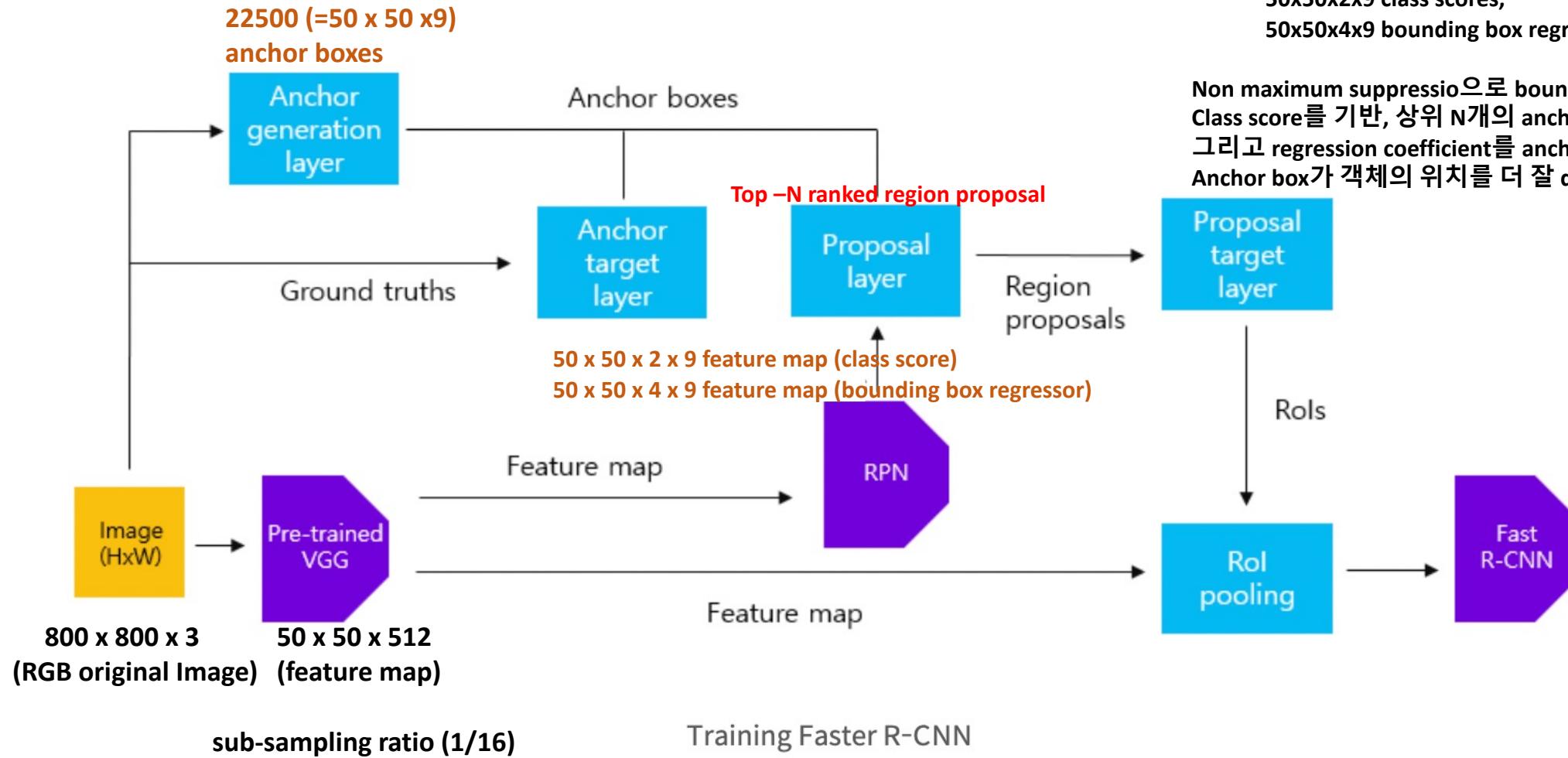
Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN



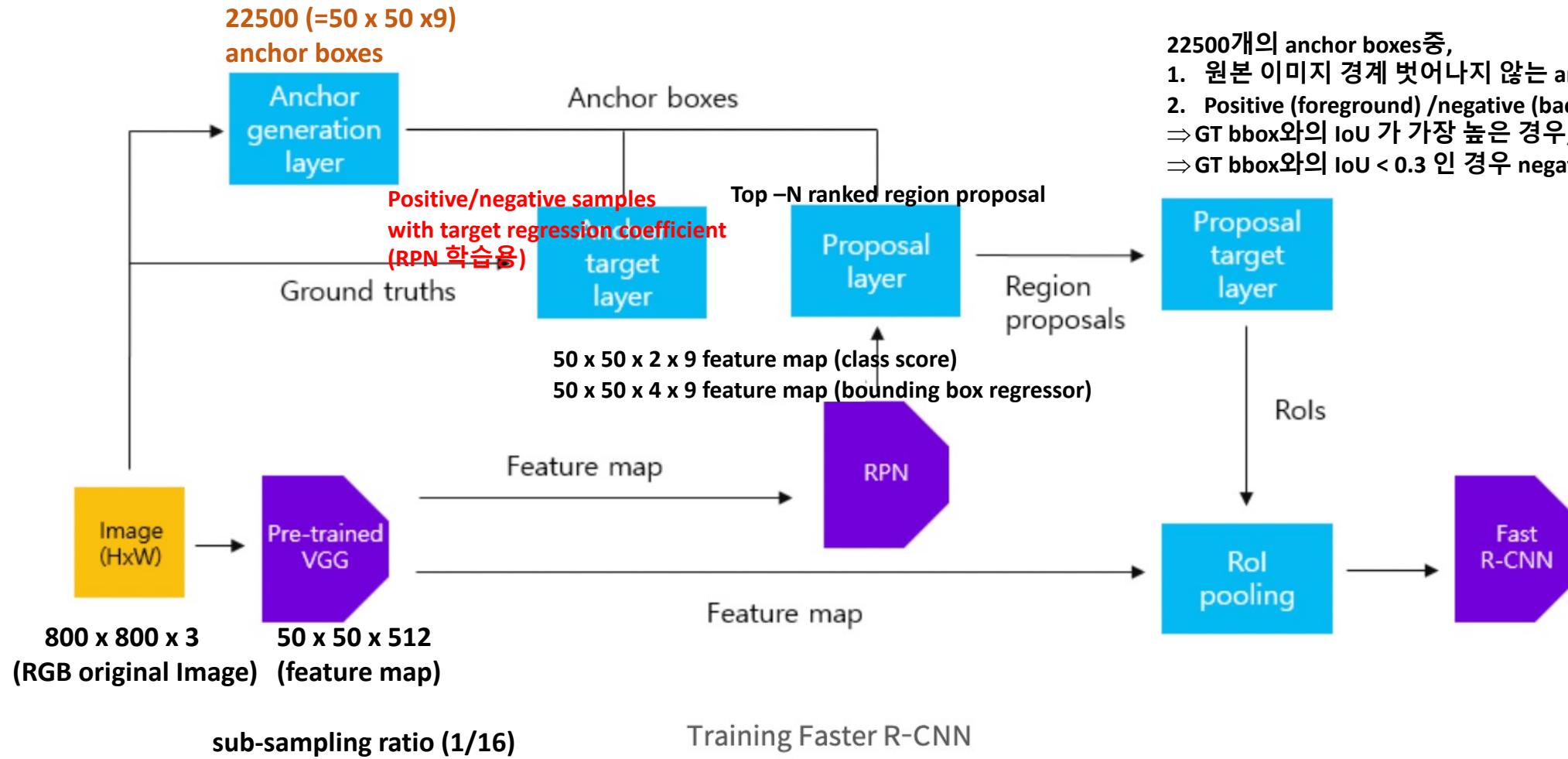
Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN



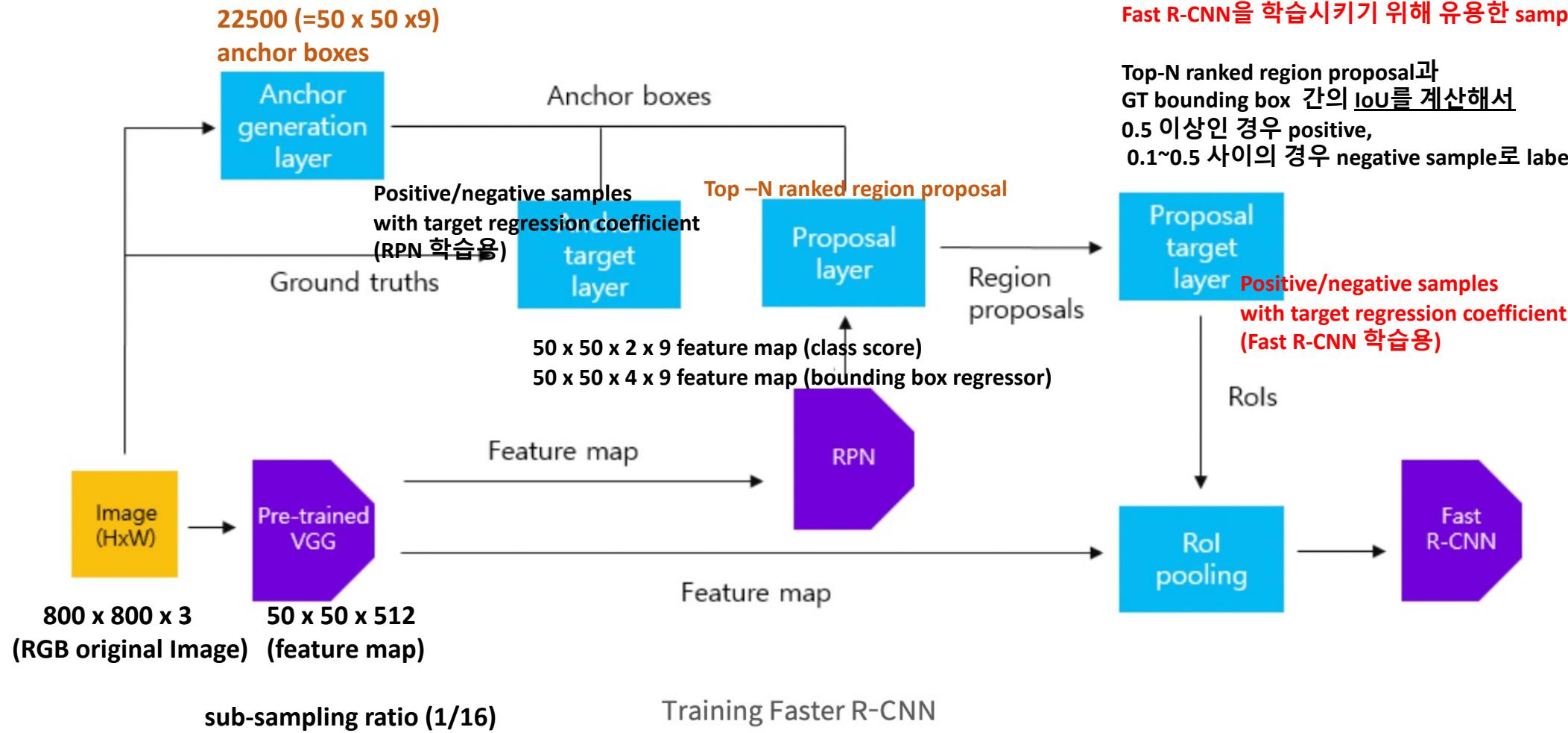
Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN



Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN



Select anchors for training Fast-RCNN by Proposal Target layer

Fast R-CNN을 학습시키기 위해 유용한 sample 선택하는 과정

Top-N ranked region proposal과
GT bounding box 간의 IoU를 계산해서
0.5 이상인 경우 positive,
0.1~0.5 사이의 경우 negative sample로 label

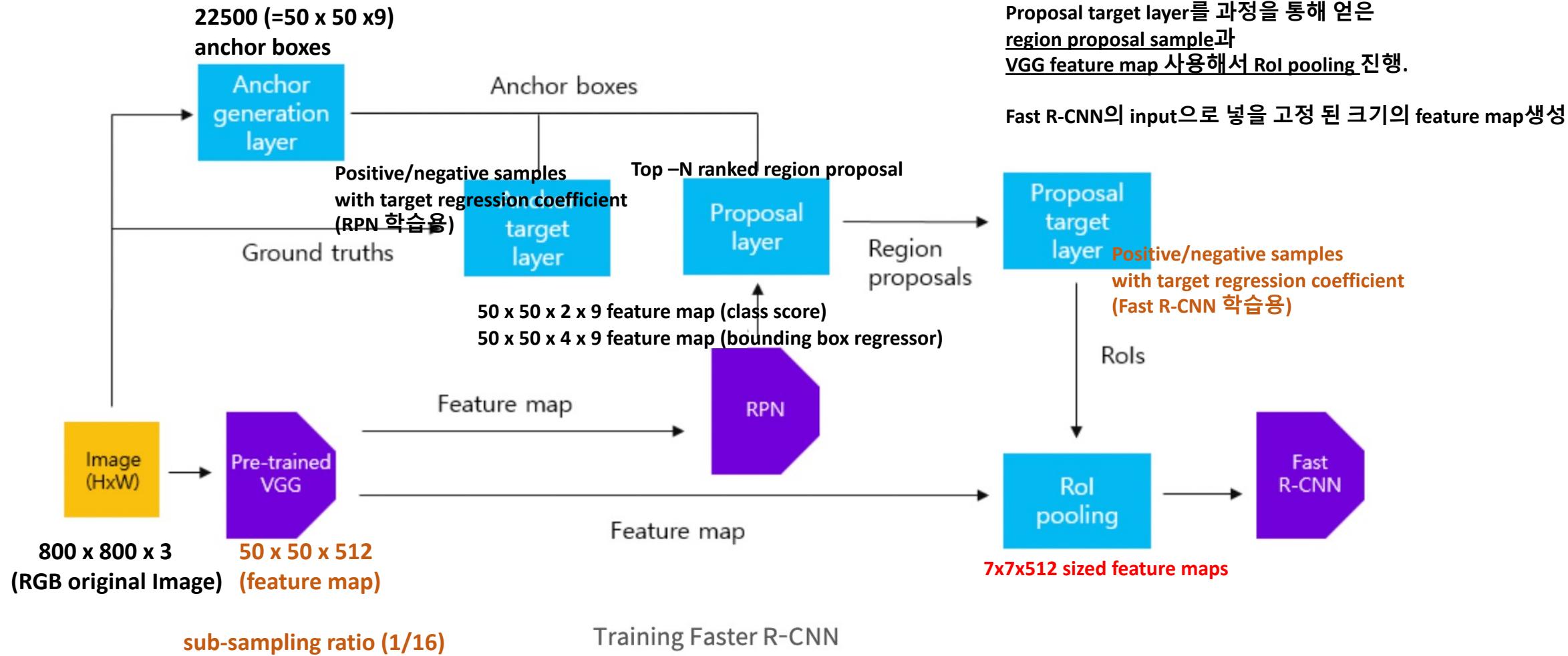
Proposal target layer
Positive/negative samples
with target regression coefficient
(Fast R-CNN 학습용)

Rols
Fast R-CNN

Training Faster R-CNN

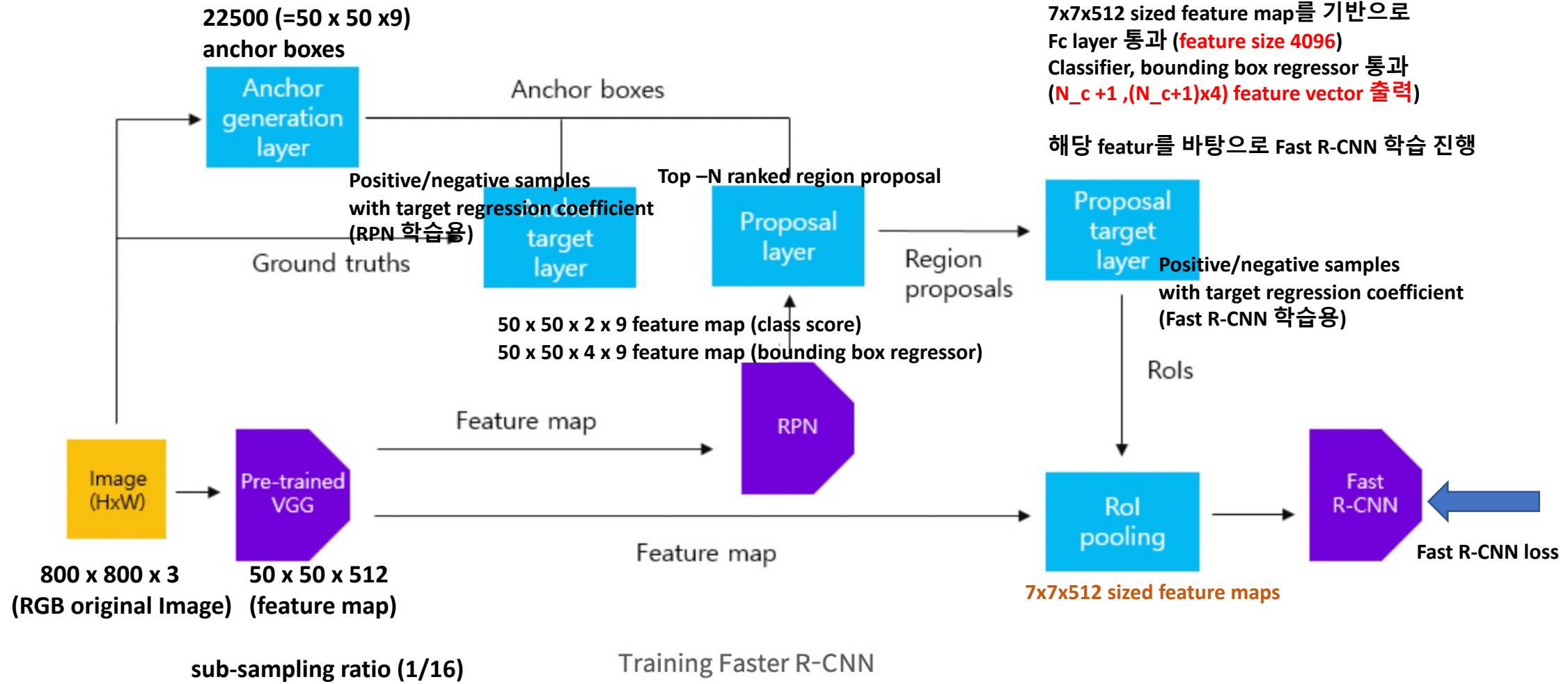
Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN



Faster R - CNN

■ Detail of Training Scheme of Faster R-CNN

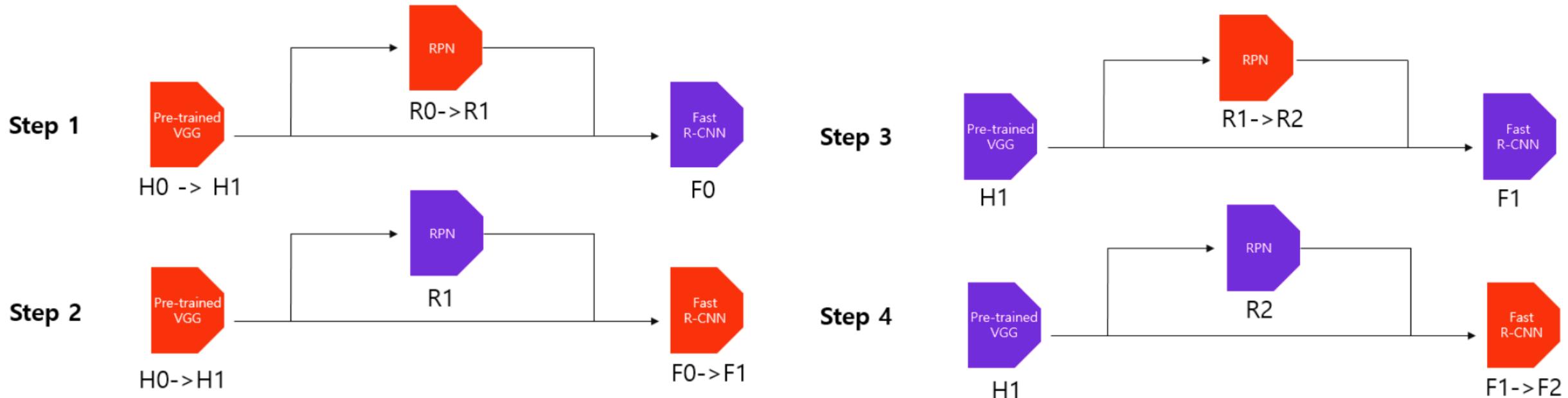


Faster R - CNN

■ How to train RPN and Fast R-CNN ?

- Alternating Training (RPN , Fast R-CNN 번갈아 가며 training)

Purple: 학습되지 않는 모듈
Red : 학습되는 모듈



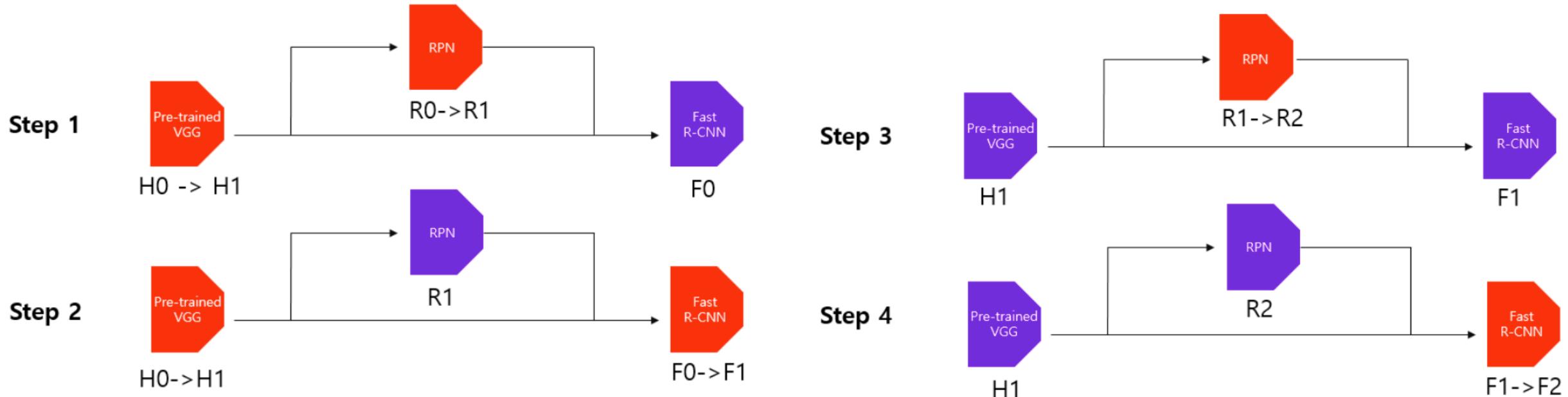
Step 1 : Anchor generation Layer에서 생성된 anchors (50x50x9) 와 Ground truth Bbox 를 이용해 positive/negative 데이터셋 구성. **RPN 학습.** Pre-trained VGG16에도 gradient가 흐름 (학습진행)

Faster R - CNN

■ How to train RPN and Fast R-CNN ?

- Alternating Training (RPN , Fast R-CNN 번갈아 가며 training)

Purple: 학습되지 않는 모듈
Red : 학습되는 모듈



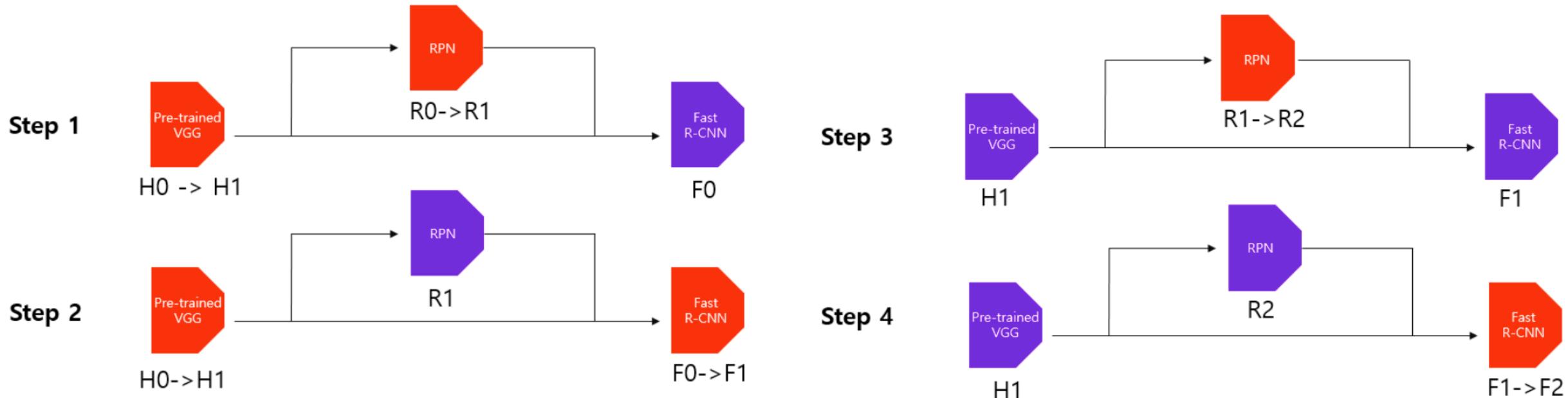
Step 2 : RPN에서 얻은 Top-N ranked region proposal와 Ground Truth Bbox를 이용해 positive/negative 데이터셋 구성. Fast R-CNN 학습. Pre-trained VGG16에도 gradient가 흐름 (학습진행)

Faster R - CNN

■ How to train RPN and Fast R-CNN ?

- Alternating Training (RPN , Fast R-CNN 번갈아 가며 training)

Purple: 학습되지 않는 모듈
Red : 학습되는 모듈



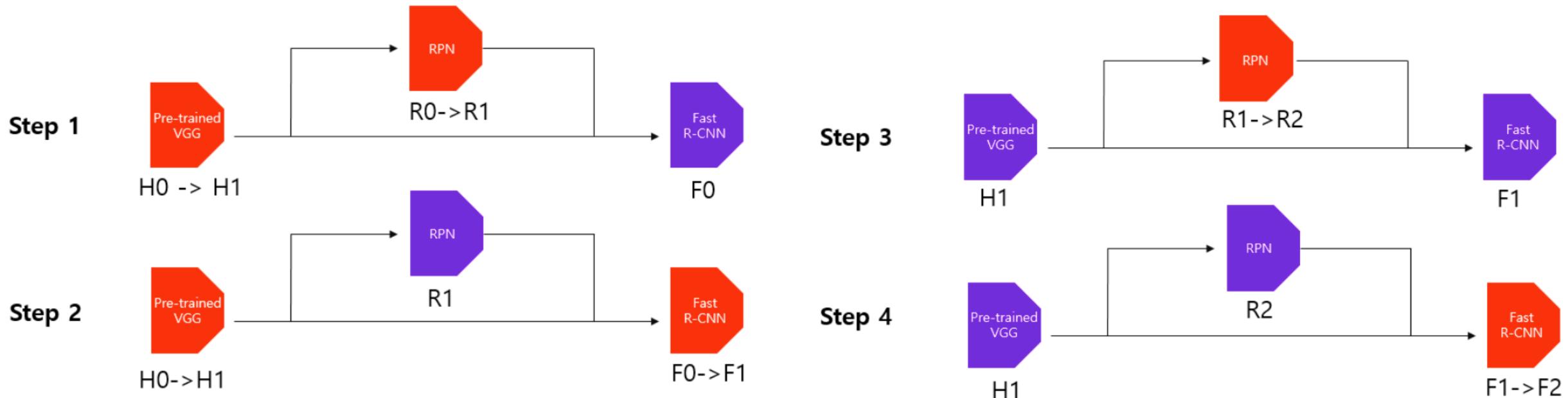
Step 3 : Anchor generation Layer에서 생성된 anchors (50x50x9) 와 Ground truth Bbox 를 이용해 positive/negative 데이터셋 구성. **RPN 학습**. Pre-trained VGG16는 Freeze

Faster R - CNN

■ How to train RPN and Fast R-CNN ?

- Alternating Training (RPN , Fast R-CNN 번갈아 가며 training)

Purple: 학습되지 않는 모듈
Red : 학습되는 모듈



Step 4 : Step 3의 RPN에서 얻은 Top-N ranked region proposal와 Ground Truth Bbox를 이용해 positive/negative 데이터셋 구성. **Fast R-CNN 학습.** Pre-trained VGG16는 Freeze

Faster R - CNN

■ How to train RPN and Fast R-CNN ?

- Alternating Training (RPN , Fast R-CNN 번갈아 가며 training)

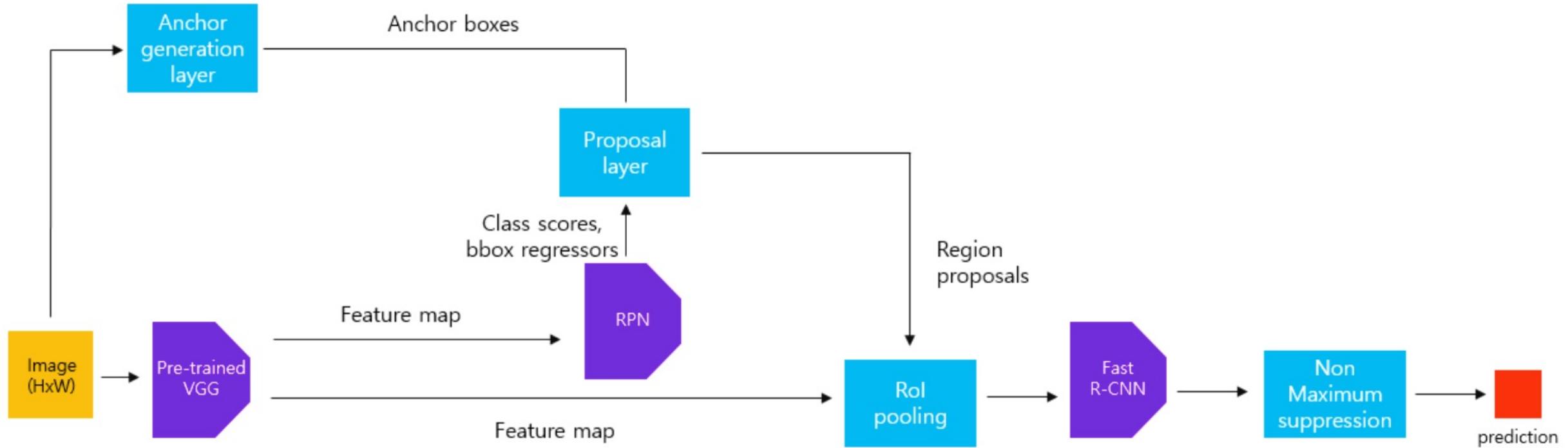
학습 방법이 필요이상으로 복잡함

⇒ 저자가 논문 마감일에 맞춰서 설계하다 보니 그렇게 됨..

(추후에 RPN과 Fast-RCNN을 동시에 학습할 수 있는 Approximated Joint Training 으로 대체됨.)

Faster R - CNN

■ Inference of Faster R – CNN (how to detect)



RPN으로부터 나온 Top-N region proposal을 이용해 detection 수행.

최종적으로 얻은 predicted box에 **Non maximum suppression**을 적용해서 최적의 bounding box만을 결과로 출력

Faster R - CNN

■ Experiments

- RPN을 사용 했을때 Selective Search에 비해서 상당한 속도향상을 보임

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 |

Faster R - CNN

■ Experiments

- Predefined Anchor은 3 scales, 3 ratios로 사용했을때 성능이 가장 잘나왔다
- Anchor를 9개로 잡은 이유.

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 |

Faster R - CNN

■ Performance

- R-CNN, Fast R-CNN, Faster R-CNN의 성능 비교
- Faster R-CNN 모델에서부터 실시간으로 object detection이 가능한 수준으로 성능이 올라감

per image

| system | time | 07 data | 07+12 data |
|--------------|-------|---------|------------|
| R-CNN | ~50s | 66.0 | - |
| Fast R-CNN | ~2s | 66.9 | 70.0 |
| Faster R-CNN | 198ms | 69.9 | 73.2 |

detection mAP on PASCAL VOC 2007, with VGG-16 pre-trained on ImageNet