Object detection: R-CNN Models

Sang Yup Lee

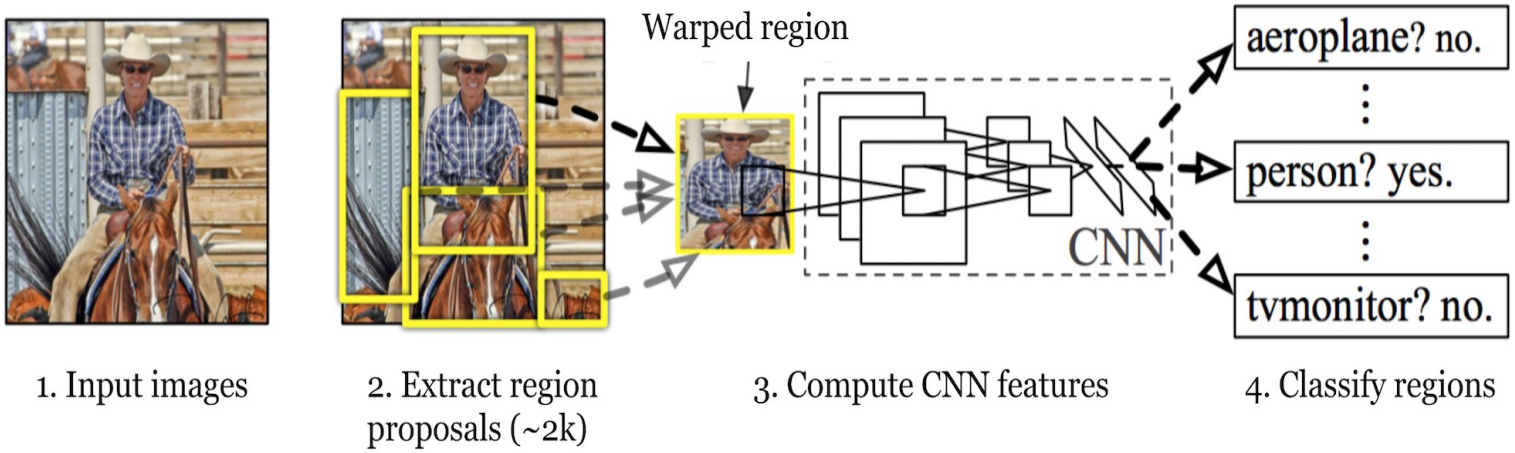
R-CNN family 소개

- Region-based convolutional neural networks
- 2 stage detectors
  - Region proposal + Detection[Classification + Localization]
- Models
  - R-CNN
    - Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).
  - Fast R-CNN
    - Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).
  - Faster R-CNN
    - Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.
  - Mask R-CNN
    - He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2016). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

R-CNN

R-CNN

- 개요
  - Proposed by Girshick et al. at UC Berkeley in 2014
  - One of the first large, successful applications of CNN to problems of object detection
  - 작동 방식



R-CNN

- 2 단계 구성
  - 단계 1: Region Proposal
    - Extract regions of interest (ROIs)
      - Selective search 방법 사용 ⇒ 2000개의 ROI 추출
        - Efficient Graph-Based Image Segmentation (Felzenszwalb & Huttenlocher, 2004) 방법을 사용해서 작은 segment 들을 찾음
        - 유사한 segments를 합쳐서 물체가 있을만한 지역을 추천
          - Color, texture, size, fill
        - 신경망 아님 (즉, learnable parameter를 갖지 않음)
        - Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International journal of computer vision, 104, 154-171.

R-CNN

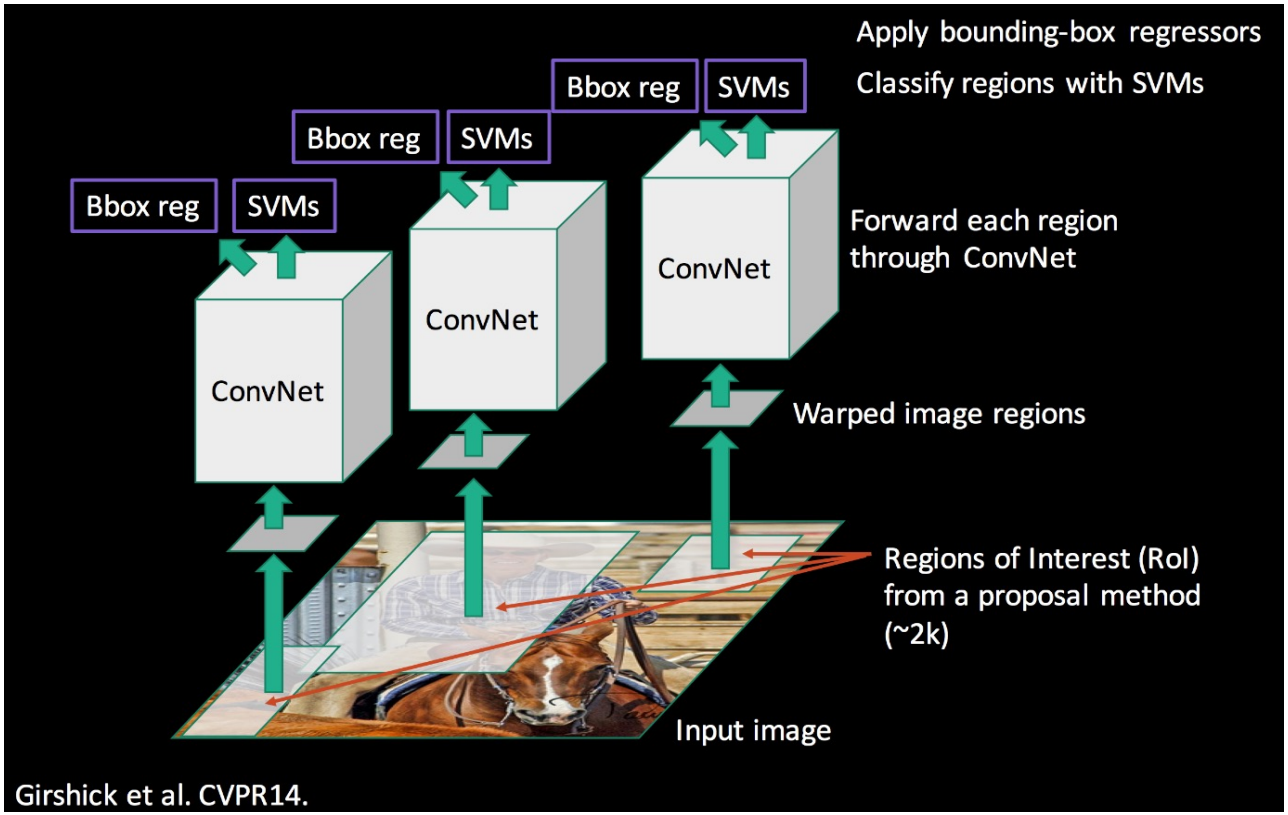
- 2 단계 구성
  - 단계 2: Object Detection
    - Feature extraction
      - 이전 단계에서 추출된 각 ROI들에 대해 CNN (AlexNet)을 적용해서 feature map 추출
      - RoI (즉, region proposal)를 AlexNet에 입력하기 위해 각 RoI의 크기를 227x227로 맞춰줌. 이를 위해 warping 방법 사용
      - AlexNet이 출력하는 feature maps 를 사용해서 classification과 localization 작업 수행
      - Classification과 localization 작업을 위해 각 feature map을 4096 차원의 벡터로 변환 ⇒ 해당 벡터에 SVMs (for classification)과 선형회귀 모형 (for localization)을 적용

R-CNN

- 단계 2: Object Detection (cont'd)
  - Classification
    - 각 feature map에 대해 SVMs 을 적용해 물체의 클래스를 예측
    - 클래스별로 하나의 (binary) SVM을 사용
  - Localization
    - 각 feature map에 선형회귀 모형을 적용해 localization 작업 수행

R-CNN

- 전체적 구조


Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.

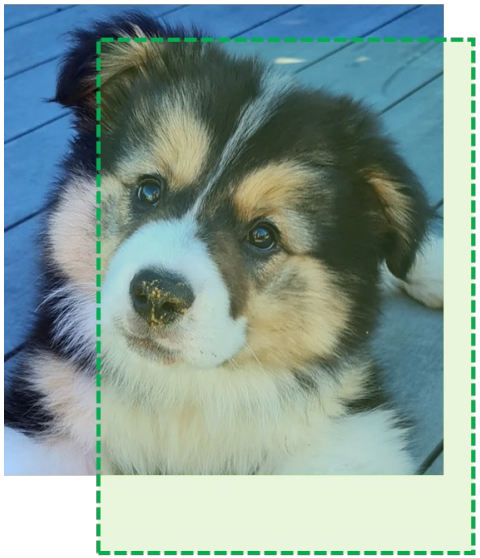
R-CNN

- 학습
  - 학습 데이터
    - PASCAL VOC (# of classes = 20)
    - ILSVRC2013 (# of classes = 200)
  - Positive proposals vs. negative proposals
    - Threshold IoU with GTBB = 0.5
      - if IoU >= 0.5, then positive, otherwise negative
  - mini-batch size = 128
    - 랜덤하게 32개의 positive regions와 96개의 negative regions 를 선택해서 미니 배치를 구성 (positive region을 상대적으로 더 많이 sampling 함)

R-CNN

- 학습 (cont'd)
  - 각 region proposal에 대해 정답 경계상자 할당
  - Classification

GTBB (Label=Dog)



<Region proposal>

해당 proposal 의 정답 레이블은 'Dog' 가 됨

- 해당 클래스에 대한 one-hot vector 가 정답 정보로 사용
- 각 클래스에 대한 확률은 binary SVM을 이용해서 예측

R-CNN

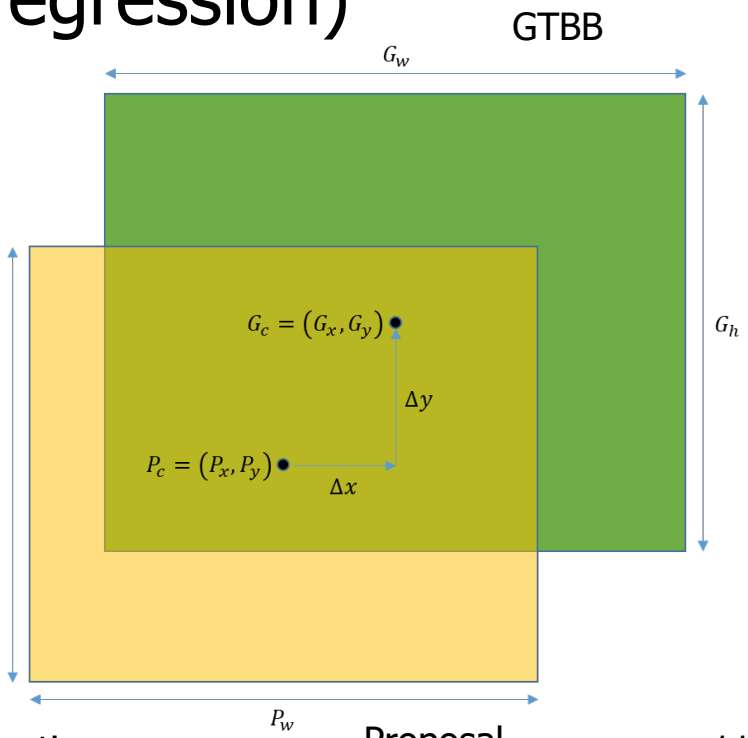
- 학습 (cont'd)
  - Localization (Bounding-box regression)
    - 선형회귀 모형을 사용
    - 모형에 입력되는 데이터
      - N 개의 training pairs

$\{(P^i, G^i)\}_{i=1,\ldots,N}$, where $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$

, where P^i_x, P^i_y는 프로포절 i의 중심 좌표 (x, y)이고, P^i_w, P^i_h는 프로포절 i의 너비와 높이 => 단위는 픽셀 단위

$$G^i = (G^i_x, G^i_y, G^i_w, G^i_h)$$

=> 이는 Ground Truth BB에 대한 정보



R-CNN

- 학습
  - Localization (cont'd)
    - 학습을 위해 네 개의 함수 사용: $d_x(P), d_y(P), d_w(P), d_h(P)$
    - 각 함수는 아래와 같이 표현

$$d_\star(P) = \mathbf{w}_\star^T \phi_5(P), \text{where } \star \in \{x, y, w, h\}$$

$\phi_5(P)$는 the pool5 features of proposal P
$\mathbf{w}_\star$는 a vector of learnable parameters => 즉 학습을 통해서 학습되는 파라미터 벡터

    - 아래 식을 이용해서 $\mathbf{w}_\star$를 계산 (아래는 L2 규제화를 사용)

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \phi_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2 . \quad (5)$$

R-CNN

- 학습
  - Localization (cont'd)
    - 위 식에서 $t_\star$는 아래 공식을 이용해서 계산

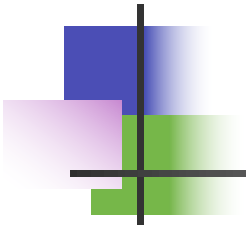
$$t_x = (G_x - P_x)/P_w$$
$$t_y = (G_y - P_y)/P_h$$
$$t_w = \log(G_w/P_w)$$
$$t_h = \log(G_h/P_h).$$

R-CNN

- 추론 (inference)
  - 좌표 예측
    - 학습된 파라미터값 이용해서 각 함수(즉, $d_\star(P)$)의 값을 계산 (아래 식 사용)
      - $d_\star(P) = \mathbf{w}_\star^T \phi_5(P)$
    - 그리고 이를 이용해서 GTBB의 좌표를 예측 (아래 식 사용)

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P)).$$

  - NMS 방법 사용



R-CNN

- 추론 (inference)
  - NMS 방법 적용

R-CNN

- 주요 단점
  - 계산량이 많아 시간이 오래 걸린다.
    - 각각의 ROI에 별도의 사전 학습모형 (AlexNet) 적용 (약 2000개)
    - 각 ROI에 대한 feature map에 SVM을 여러번 적용
  - 여러 개의 구성 요소들
    - Selective search, CNN Feature Extractor, SVM 과 Bounding box regressor 로 구성되어 복잡

Fast R-CNN

Fast R-CNN

- 개요
  - Proposed in 2015, mainly to address the drawbacks of R-CNN
  - R-CNN의 주요 한계
    - 이미지에 selective search를 이용해서 2000개의 region proposals를 추출
    - 각 proposal에 CNN 기반 사전 학습 모형 (AlexNet)을 적용해서 4096 차원의 피쳐맵을 추출
    - 여기에 SVM을 적용해서 클래스별로 예측
    - Localization은 선형회귀모형을 이용해서 좌표를 예측
    - 2000개의 region proposal 각각에 CNN을 적용하는 것이 큰 한계 ⇒ 속도가 너무 오래 걸림
    - 여러 단계의 작업을 수행 ⇒ 번거로움

Fast R-CNN

- 주요 특성
  - 이미지에 CNN 모형 (VGGNet)을 직접 적용해 하나의 feature map을 추출
  - 이미지에 selective searc를 적용해서 RoIs 추출 (2000개)
  - 각 RoI를 feature map에 매핑 (mapping)
  - 각 mapped RoI를 동일한 크기로 만들기 위해 RoI pooling 수행
    - fully connected layer에 입력하기 위해서는 동일한 크기로 맞춰주는 것이 필요
  - 두 종류의 fully connected layer 적용
    - For classification: 각 RoI에 대해서 K+1개의 확률값 출력 (K=# of classes), 이를 위해 소프트맥스 함수 사용
    - For bounding box regression: 각 RoI에 대해, 각 클래스별로 4개의 좌표값을 출력 (즉, 각 ROI 마다 4x20개의 값을 출력)

Fast R-CNN

- 주요 특성 (cont'd)
  - Multi-task loss 사용
    - 학습에서 사용된 각 ROI는 정답 레이블과 정답 offset 값을 갖음
    - 각 RoI별 loss: $L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v)$
      - $L_{cls}(p, u)$ 는 교차 엔트로피
      - u = ground truth class, v = ground-truth bounding-box regression target
      - 백그라운드 클래스에 대해서는 localization loss는 계산하지 않음
      - $\lambda = 1$

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i), \qquad (2)$$

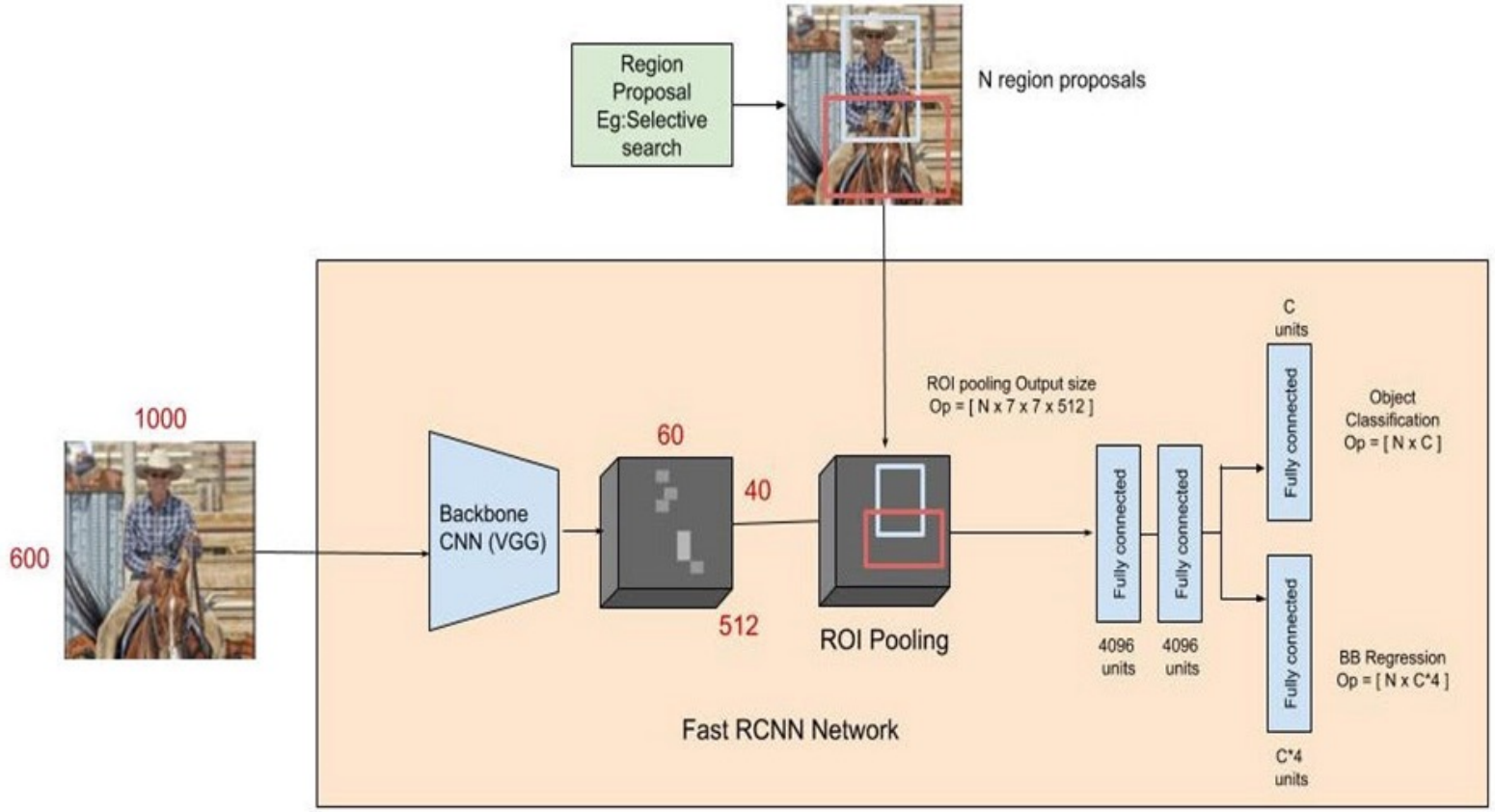
in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

  - Training is single-stage

Fast R-CNN

- 구조


Region Proposal Eg:Selective search
N region proposals
1000
600
Backbone CNN (VGG)
60
40
512
ROI pooling Output size
Op = [N x 7 x 7 x 512]
ROI Pooling
Fully connected
Fully connected
4096 units
4096 units
C units
Fully connected
Object Classification
Op = [N x C]
Fully connected
C*4 units
BB Regression
Op = [N x C*4]
Fast RCNN Network

Fast R-CNN

- RoI pooling
  - SPPNet (Spatial pyramid pooling networks)의 방법을 사용
    - He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904-1916.
  - 하지만 pyramid 구조는 사용하지 않음
  - 작동 방식
    - a x a 형태의 feature map에 대해서 pooling을 적용해서 n x n 형태의 결과물을 얻고자 하는 경우
    - 필터의 크기 (가로 또는 세로의 길이) = $\lceil a/n \rceil$
    - stride = $\lfloor a/n \rfloor$
    - 여기서 $\lceil \cdot \rceil$ 와 $\lfloor \cdot \rfloor$ 는 ceiling and floor operations
      - 예) ceiling operation: 4/3 = 2, floor operation: 4/3 = 1

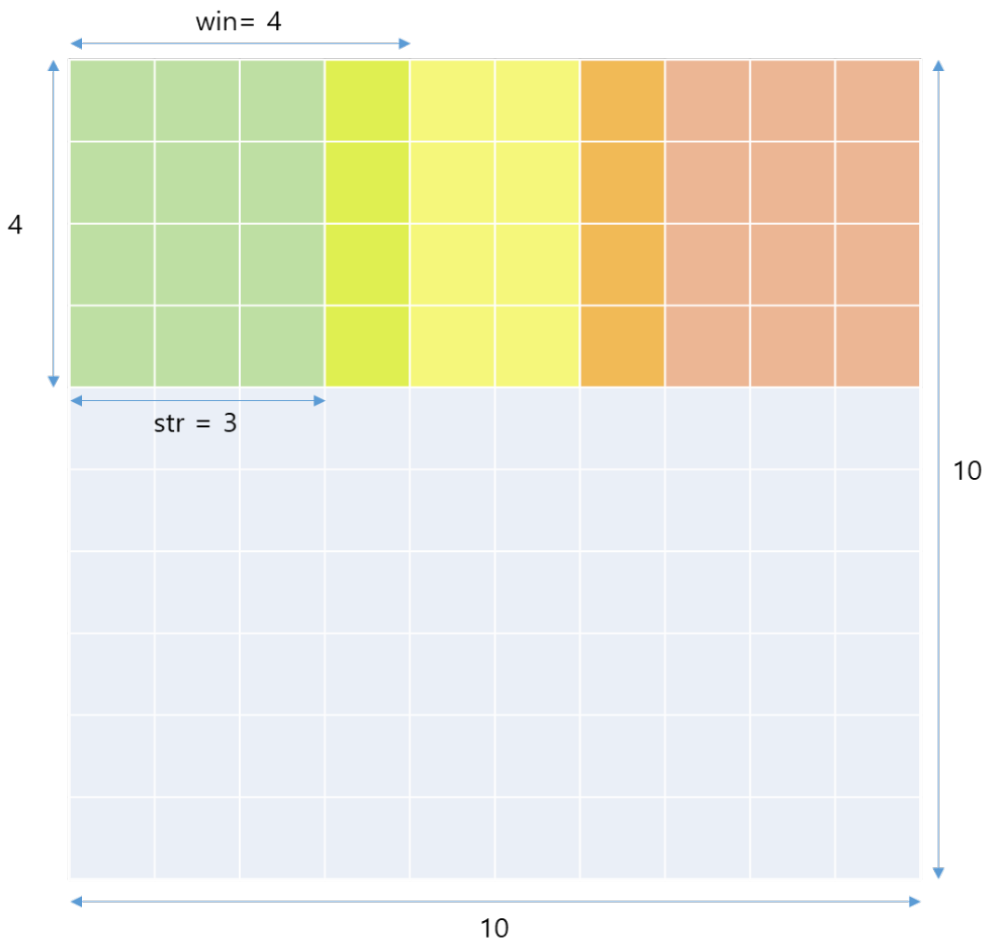
Fast R-CNN

- RoI pooling (cont'd)
  - Example) a = 10, n = 3

    $\lceil a/n \rceil = \lceil 10/3 \rceil = 4$

    $\lfloor a/n \rfloor = \lfloor 10/3 \rfloor = 3$

  - 즉, 필터의 크기는 4x4이고 스트라이드=3 ⇒ 이를 그림으로 그려 보면 오른쪽과 같음
  - 각 필터를 이용해서 max pooling 적용
  - n x n 의 결과가 얻어짐 (이를 채널별로 실행)



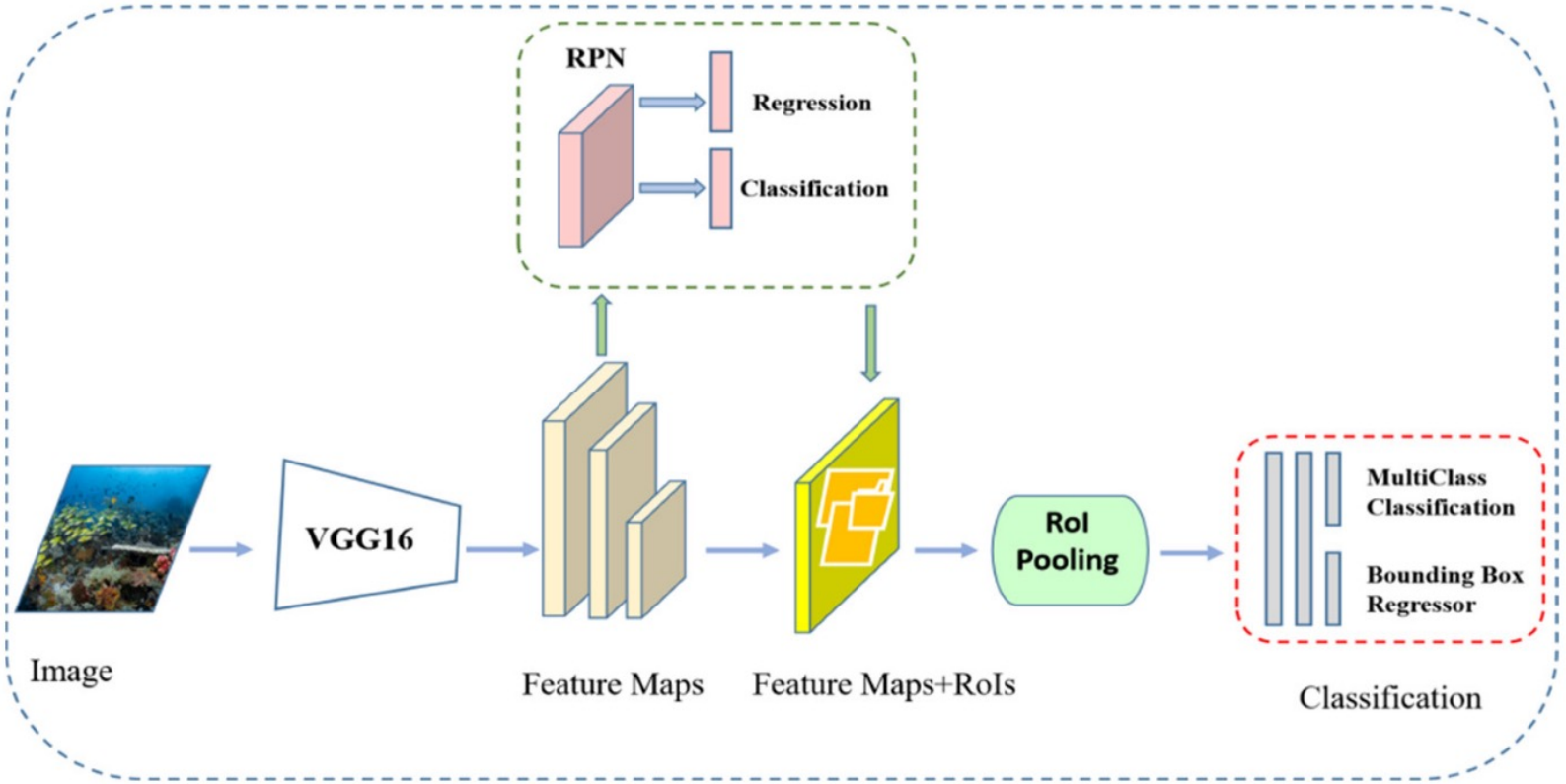
Faster R-CNN

Faster R-CNN

- Fast R-CNN의 주요 단점
  - 여전히 selective search 방법 사용 ⇒ 속도가 느리다!
- Faster R-CNN
  - 모형 구조: 두 개의 모듈로 구성
    - RPN (Region proposal networks) + Fast R-CNN
    - RPN: 신경망 기반
  - 주요 단계
    - 단계1: Region proposal network ⇒ ROIs 추출
    - 단계2: 각 ROI에 대해서 Fast R-CNN을 이용해서 classification 과 localization 작업 수행

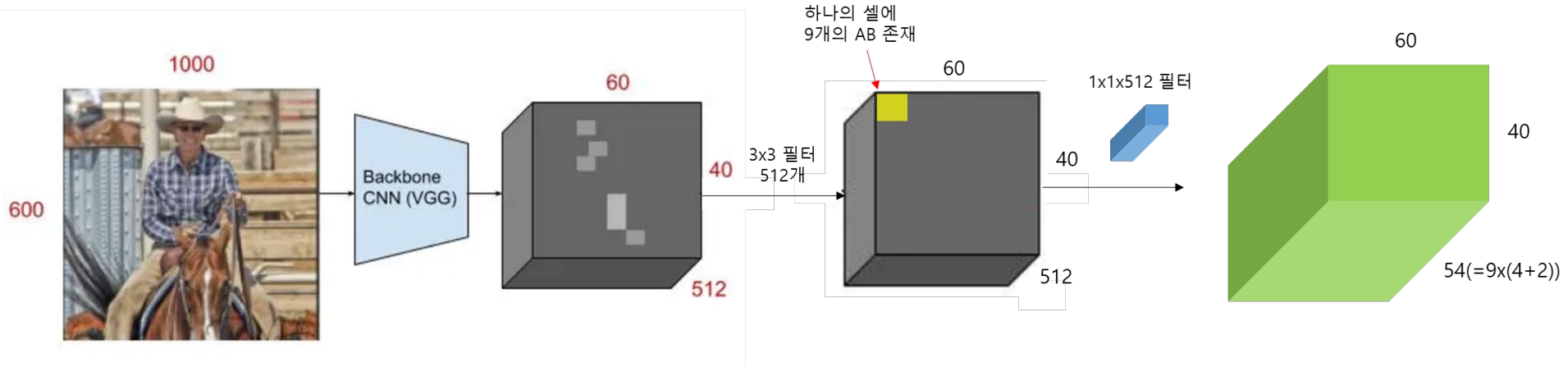
Faster R-CNN

- 모형의 구조



Faster R-CNN

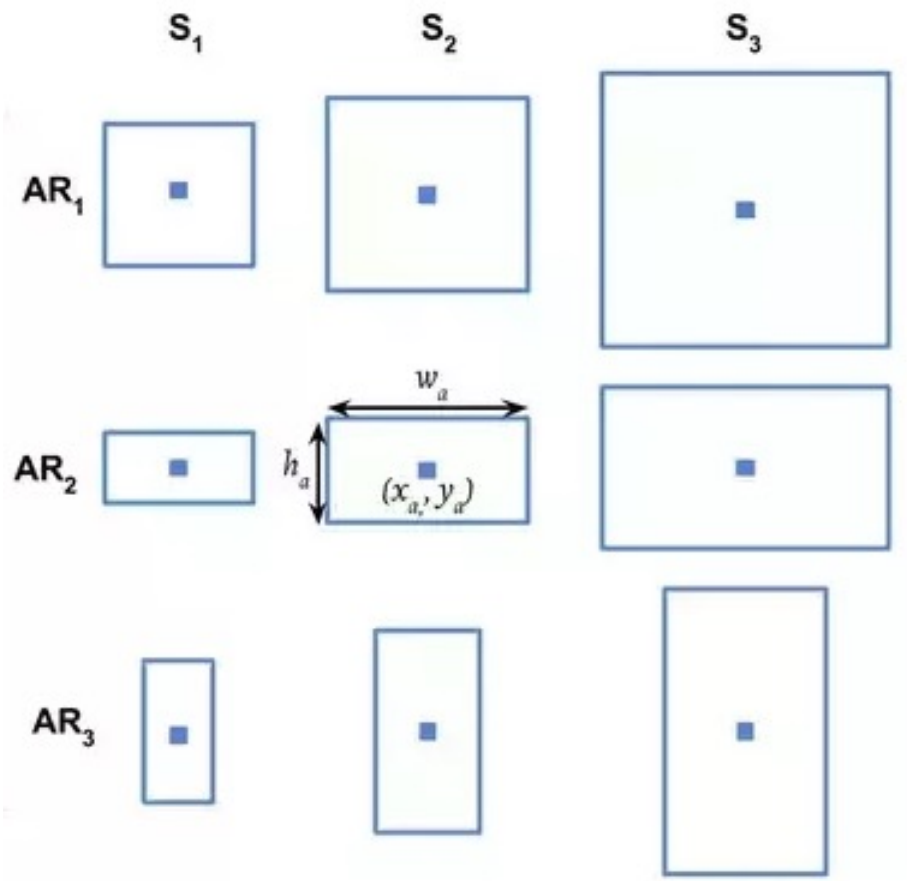
- RPN의 구조



- 각 셀마다 9 개의 anchor box 존재
- 각 AB에 대해서 4개의 좌표 (정확하게는 두 개의 중심 좌표와 너비, 높이)와 2 개의 objectness scores 출력 (즉, 물체가 있을 확률과 없을 확률) → 이러한 경우에는 softmax 활성화 함수 사용
- 하지만 sigmoid 함수를 이용해서 하나의 값(즉, 물체가 있을 확률)만 출력하는 것도 가능

Faster R-CNN

- Anchor boxes



- 3 scales: 128, 256, and 512 (pixels)
- 3 aspect ratios of 1:1, 1:2, and 2:1

Faster R-CNN

- RPN 학습
  - 각 AB 가 관측치
  - 정답 label 설정
    - positive box (물체가 있는것) vs. negative box (물체가 없는 것)
      - 물체의 클래스는 중요하지 않음
    - 두 종류의 AB에 positive label 부여
      - the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box
      - an anchor that has an IoU overlap higher than 0.7 with any ground-truth box
    - AB with negative label
      - IoU가 0.3 보다 작은 AB들
    - 다른 ABs는 학습에 사용하지 않음

Faster R-CNN

- RPN 비용함수

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

  - 여기서 i는 미니배치에 있는 AB의 인덱스, p_i는 AB i에 대해 모형을 통해 예측된 해당 AB가 물체를 갖고 있을 확률 p_i^*는 정답 레이블 => AB가 positive 이면 즉, 물체를 갖고 있으면 1 그렇지 않으면 0
  - t_i는 예측된 4 개의 좌표값에 대한 벡터, t_i^*는 정답 경계상자의 좌표값에 대한 벡터
    - 비용함수에서 사용되는 좌표를 계산하는 방식은 R-CNN과 동일
  - L_{cls}는 교차엔트로피 (즉, log loss)
  - L_{reg}는 smooth L1 비용함수
  - $p_i^* L_{reg}$ => regression loss는 오직 positive box에 대해서만 계산한다는 것을 의미
  - N_{cls}는 미니배치 크기 (N_{cls}=256)
  - N_{reg}는 feature map에 있는 cell의 수 (N_{reg}=2,400)
  - λ = 10 => Thus, both cls and reg terms are roughly equally weighted.

Faster R-CNN

- Sampling (미니배치 구성)
  - Randomly choose 256 anchor boxes
  - # of pos boxes : # of neg boxes = 1:1
  - If # of pos boxes < 128, then more neg boxes are used.
- RPN 구현 코드 (참고)
  - https://github.com/kentaroy47/frcnn-from-scratch-with-keras/blob/master/keras_frcnn/vgg.py
    - rpn 부분 참고

Faster R-CNN

- 학습 방법
  - Alternating training: RPN과 Fast R-CNN을 교차로 학습
  - 공통된 부분 (VGG의 13개 층)의 파라미터 공유를 위해 아래와 같은 4단계 학습 방법 사용
    - 단계1: RPN 학습 (앞 슬라이드의 비용함수 사용)
      - 공유된 층은 사전학습 모형의 파라미터를 사용하여 초기화하고 새로운 층은 정규 분포를 이용해서 랜덤하기 초기화 한 다음 region proposal task를 이용해서 미세조정
    - 단계2: Fast R-CNN 부분을 단계1에서 제안된 ROIs를 이용해서 학습
      - 공유되는 부분 즉, 13개 층은 사전학습 모형을 이용해서 초기화, 그리고 나머지 부분은 정규분포를 이용해서 랜덤하게 초기화
      - 여기까지는 아직 파라미터 공유가 이뤄지지 않음
    - 단계3: Detection 학습 결과로 도출된 파라미터 값을 이용해서 공유된 파라미터 초기화, 이를 이용해서 RPN 학습
      - 하지만, 학습시 공유된 파라미터의 값은 업데이트 하지 않음
      - RPN 부분의 파라미터만 업데이트
    - 단계4: 공유된 부분은 고정하고, Fast R-CNN의 고유한 부분만 classification과 localization 작업을 이용해서 미세 조정

Faster R-CNN

- 모형의 성능

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**