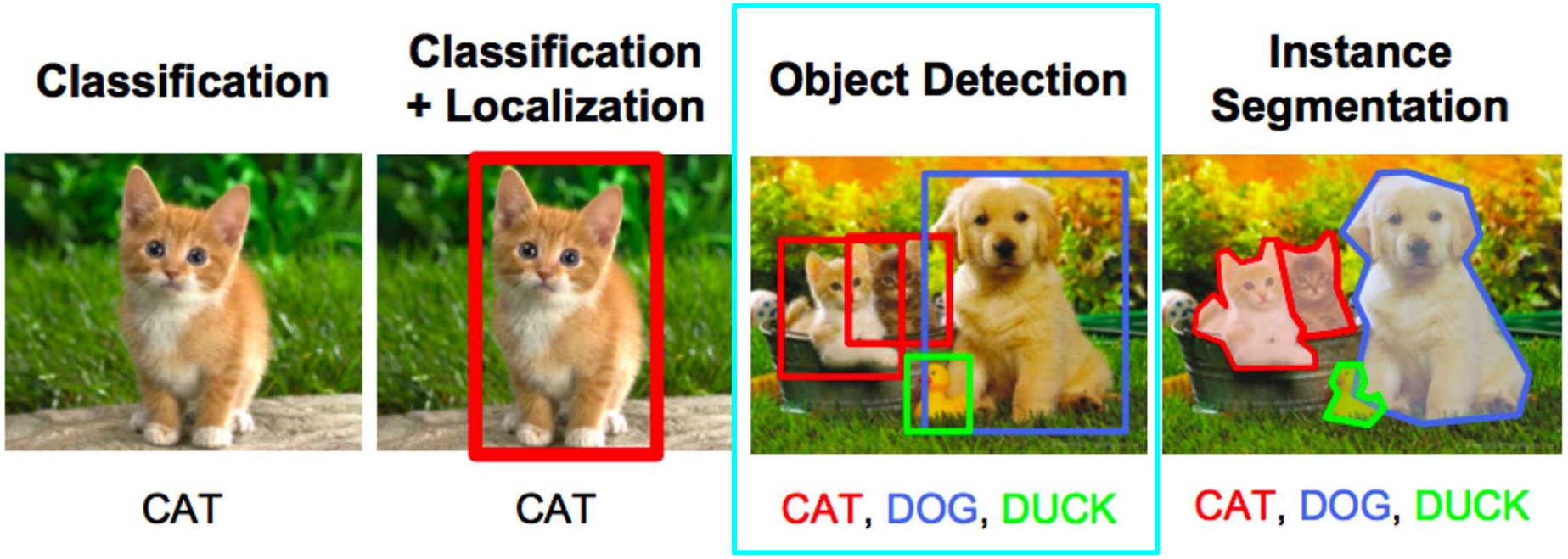
R-CNN

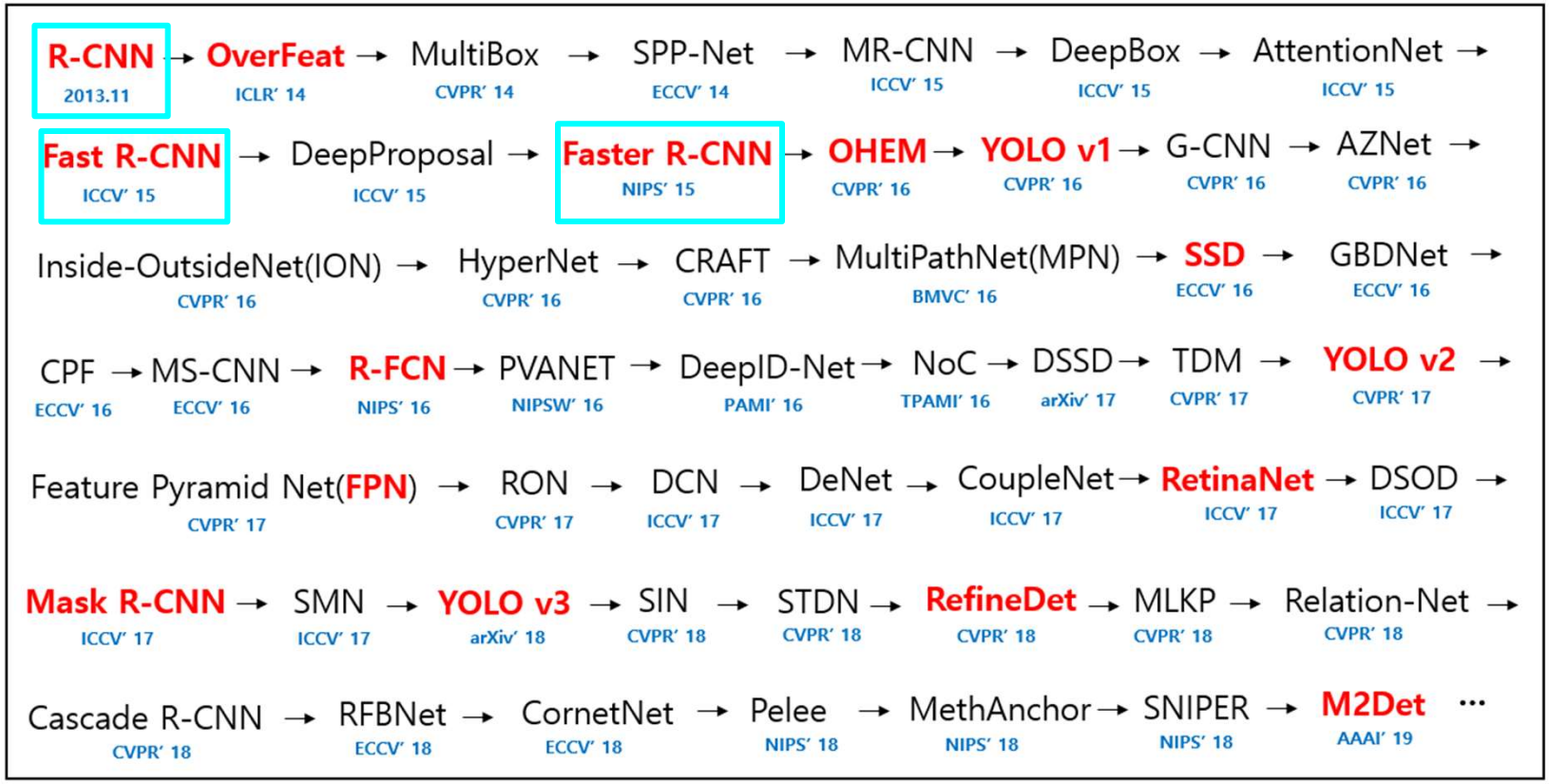
2020.04.23

이종호

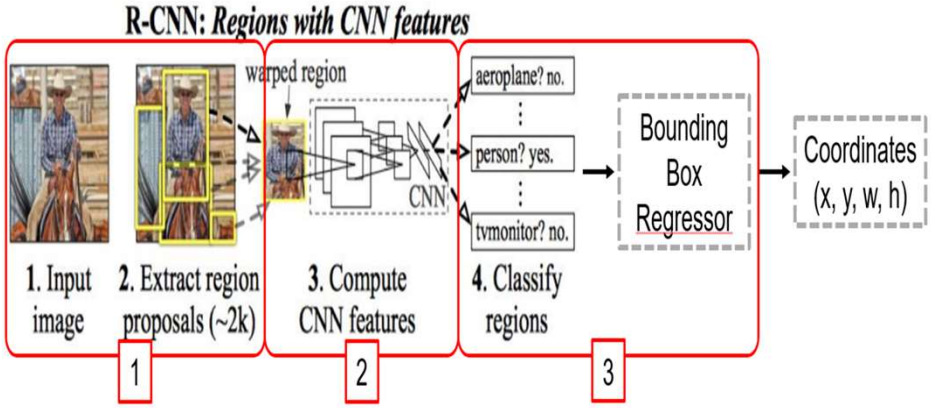
Object Detection



Object Detection

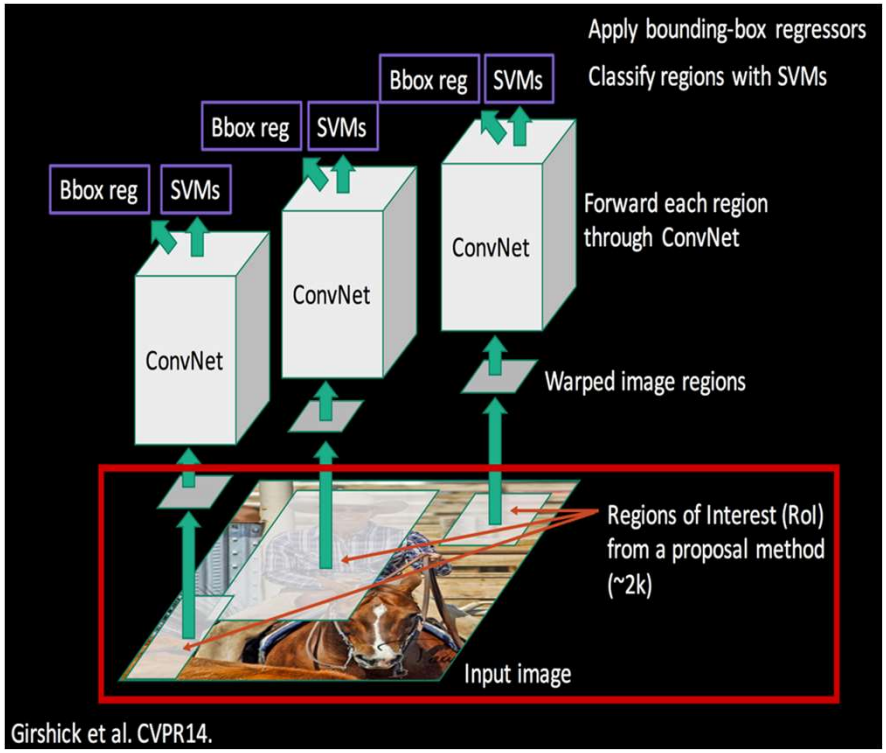


R-CNN

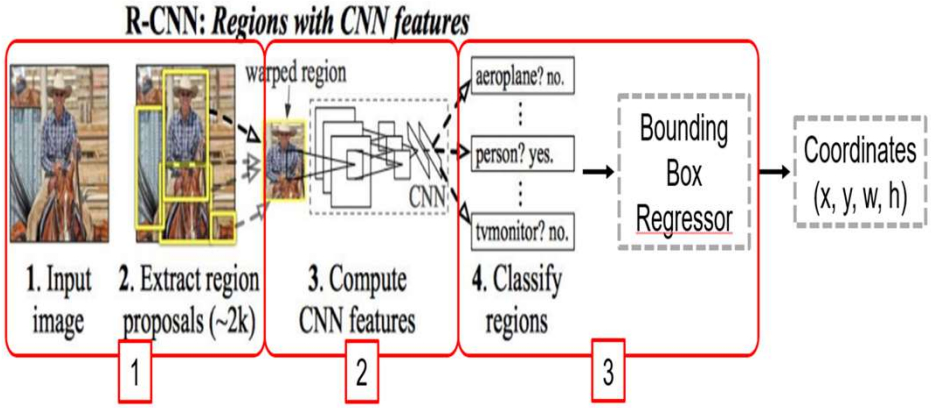


Hypothesize Bounding Boxes (Proposals)

- Image로부터 Object가 존재할 적절한 위치에 Bounding Box Proposal (Selective Search)
- 2000개의 Proposal이 생성됨.

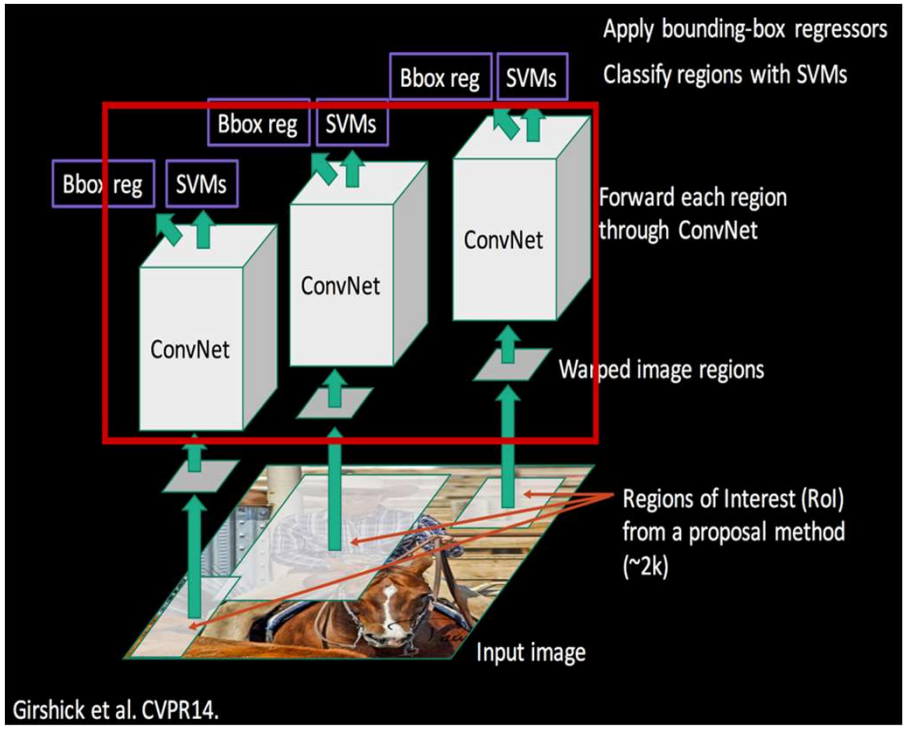


R-CNN

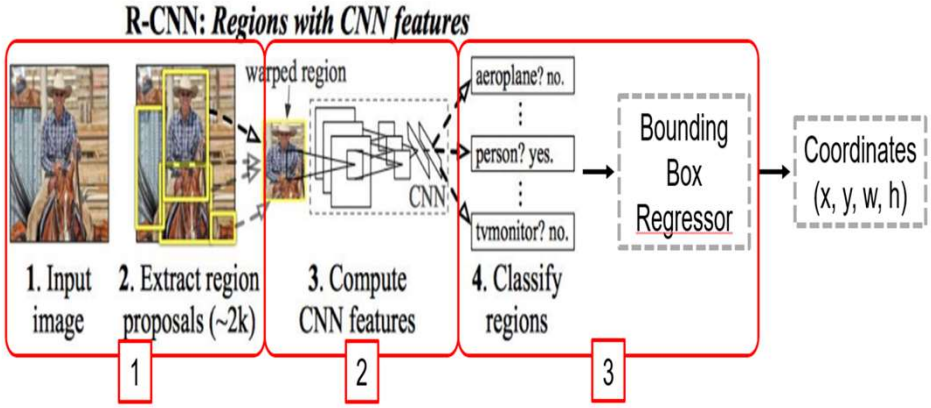


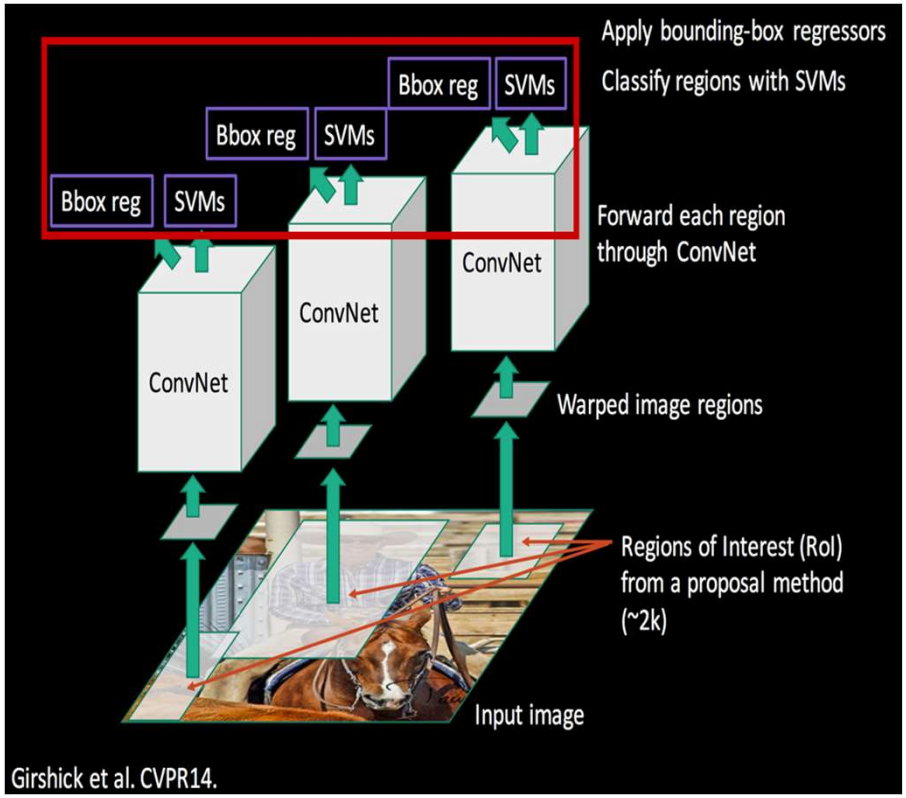
Resampling pixels / features for each boxes

- 모든 Proposal을 Crop 후 동일한 크기로 만듦



R-CNN





Classifier / Bounding Box Regressor

- 위의 영상을 Classifier와 Bounding Box Regressor로 처리

R-CNN 특징

R-CNN은 Object Detection 방법들에 비해 굉장히 뛰어난 성능을 보여준것은 분명하지만

다음과 같은 단점들이 있습니다.

1. **오래걸린다.**
    - Selective Search에서 뽑아낸 2000개의 이미지들을 CNN모델에 다 돌려서 오래걸립니다.
    - 또한 Selective Search가 CPU를 사용하는 알고리즘이기 때문에 오래걸립니다.
2. **복잡하다.**
    - R-CNN은 Multi-Stage Training을 수행하여 CNN, SVM, Bounding Box Regression까지 총 3가지 모델을 필요로 합니다.
3. **Back Propagation이 안된다.**
    - R-CNN은 Multi-Stage Training을 수행하기 때문에 SVM, Bounding Box Regression에서 학습한 결과가 CNN을 업데이트 시키지 못합니다.

Fast R-CNN

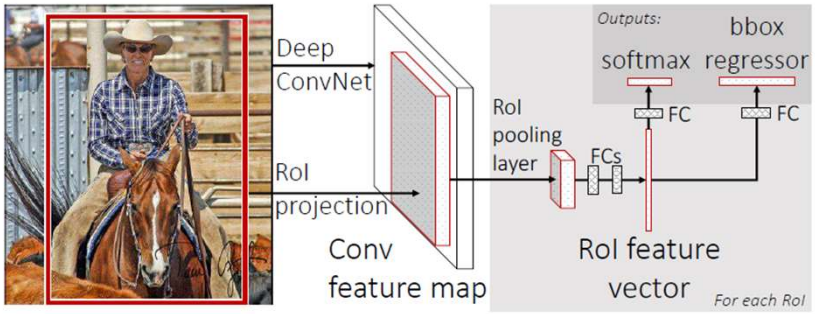
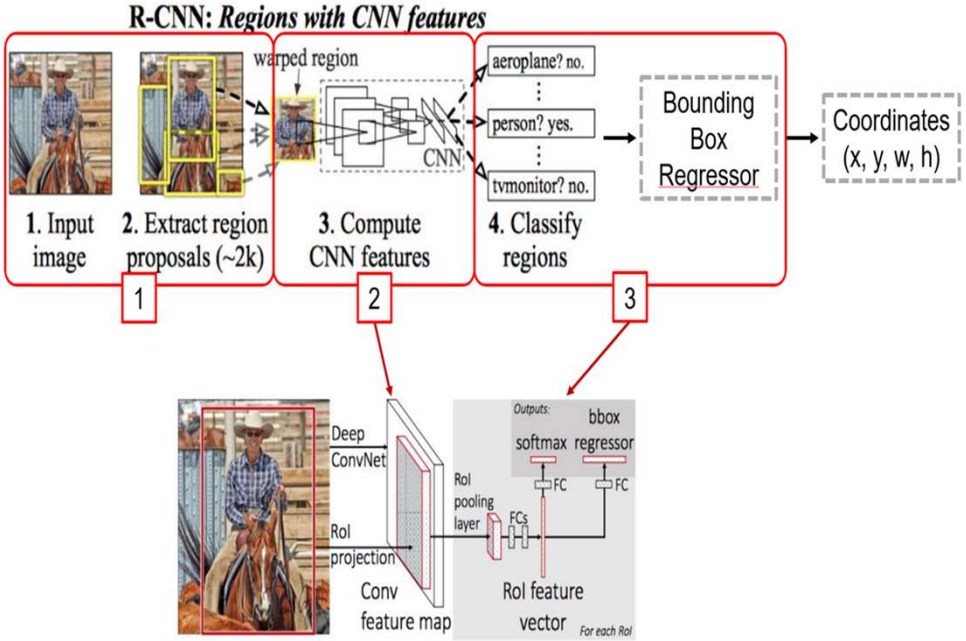
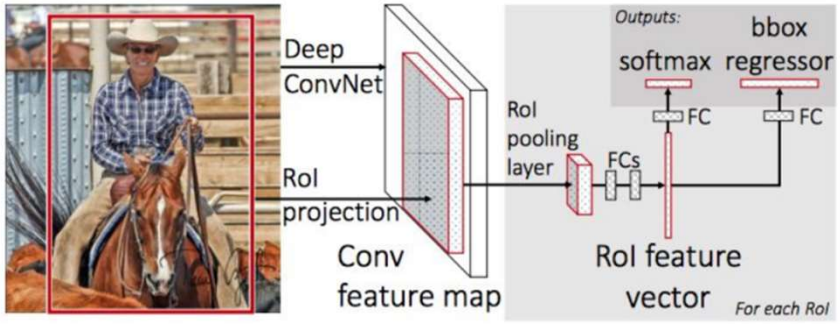
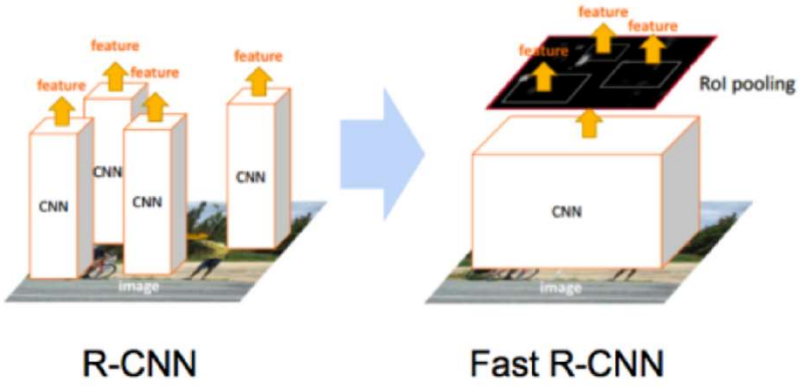


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.



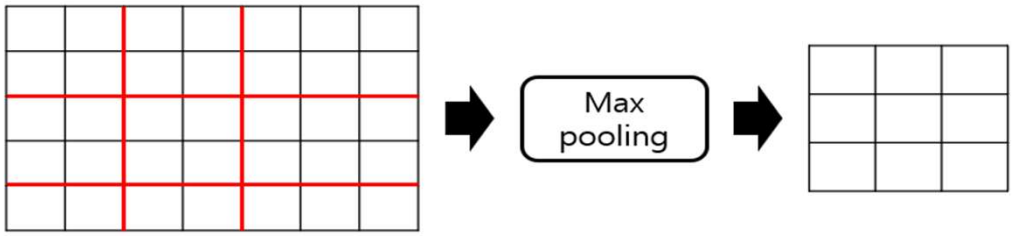
Fast R-CNN





- Fast R-CNN은 모든 Proposal이 네트워크를 거쳐야 하는 R-CNN의 병목(bottleneck)구조의 단점을 개선하고자 제안 된 방식
- 가장 큰 차이점은, 각 Proposal들이 CNN을 거치는것이 아니라 전체 이미지에 대해 CNN을 한 번 거친 후 출력 된 특징 맵(Feature map)단에서 객체 탐지를 수행

Fast R-CNN



RoI pooling layer 예시

- RoI pooling layer는 Conv를 통해 생성된 feature map에서 유효한 RoI 특징을 저차원으로 매핑하기 위해 H*W로의 max pooling을 사용합니다. 여기서 H와 W는 hyperparameter입니다. 논문에서 RoI는 직사각형 모양을 띄며 (r,c,h,w)의 튜플 형태로 정의됩니다

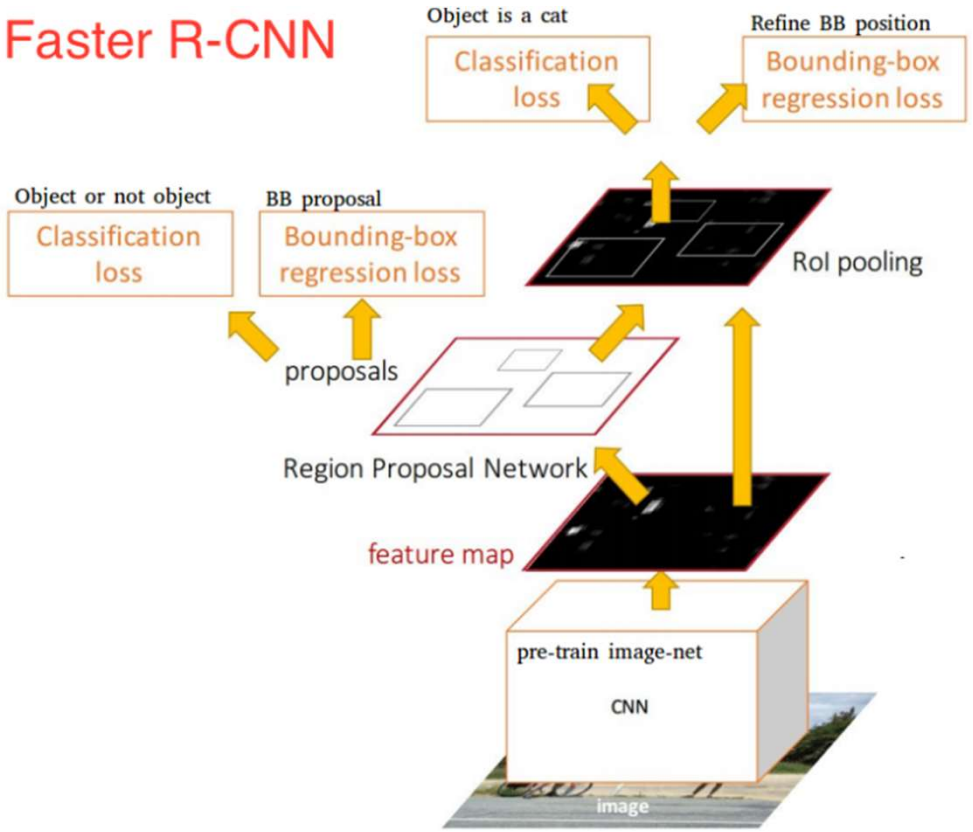
Fast R-CNN과 R-CNN 비교

- R-CNN
  - Extract image regions
  - 1 CNN per region(2000 CNNs)
  - Classify region-based features
  - Complexity: ~224 x 224 x 2000
- Fast R-CNN
  - 1 CNN on the entire image
  - Extract features from feature map regions
  - Classify region-based features
  - Complexity: ~600 x 1000 x 1
  - ~160x faster than R-CNN
- 하지만 Fast R-CNN에서 Region Proposal을 CNN Network가 아닌 Selective search 외부 알고리즘으로 수행하여 병목현상 발생

Fast R-CNN 특징

- 같은 **image**의 **proposal**들이 **convolution layer**를 공유
- **ROI Pooling** 도입
- 전체 **network**이 **End-to-end**로 한 번에 학습
- **R-CNN**보다 빠르고 더 정확한 결과

Faster R-CNN



- RPN + Fast R-CNN
- R-CNN에서는 3가지 모듈 (region proposal, classification, bounding box regression)을 각각 따로 따로 수행한다.

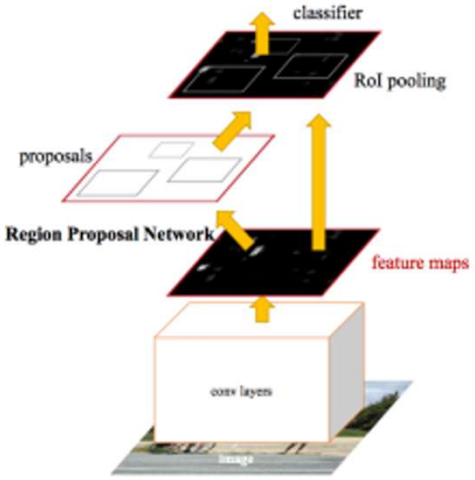
(1)region proposal 추출 → 각 region proposal별로 CNN 연산 → (2)classification, (3)bounding box regression

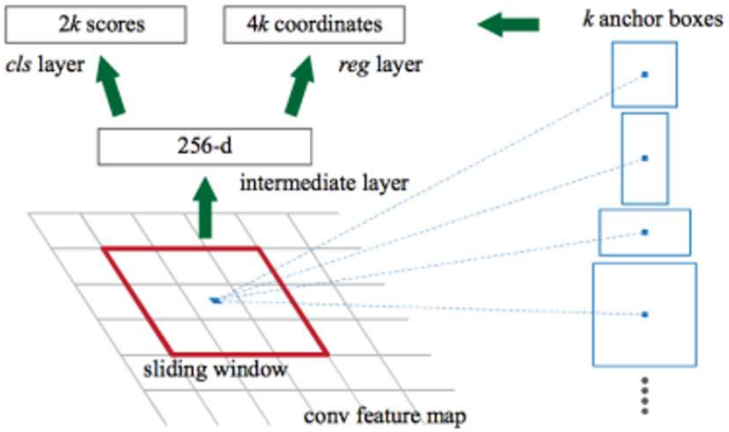
- Fast R-CNN에서는 region proposal을 CNN level로 통과시켜 classification, bounding box regression을 하나로 묶었다.

(1)region proposal 추출 → 전체 image CNN 연산 → RoI projection, RoI Pooling → (2)classification, bounding box regression

- Faster R-CNN은 Fast R-CNN구조에서 conv feature map과 RoI Pooling사이에 RoI를 생성하는 Region Proposal Network가 추가된 구조이다.

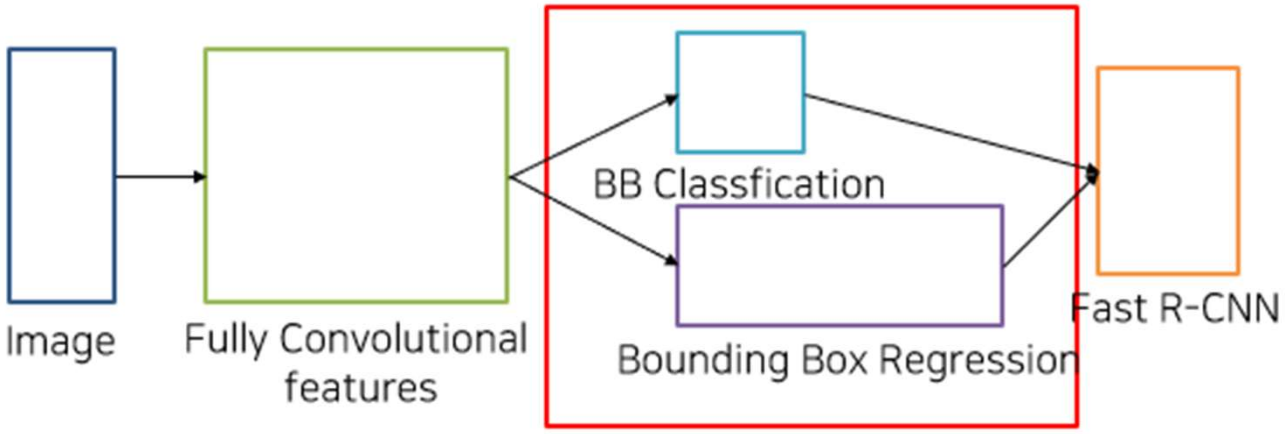
Faster R-CNN





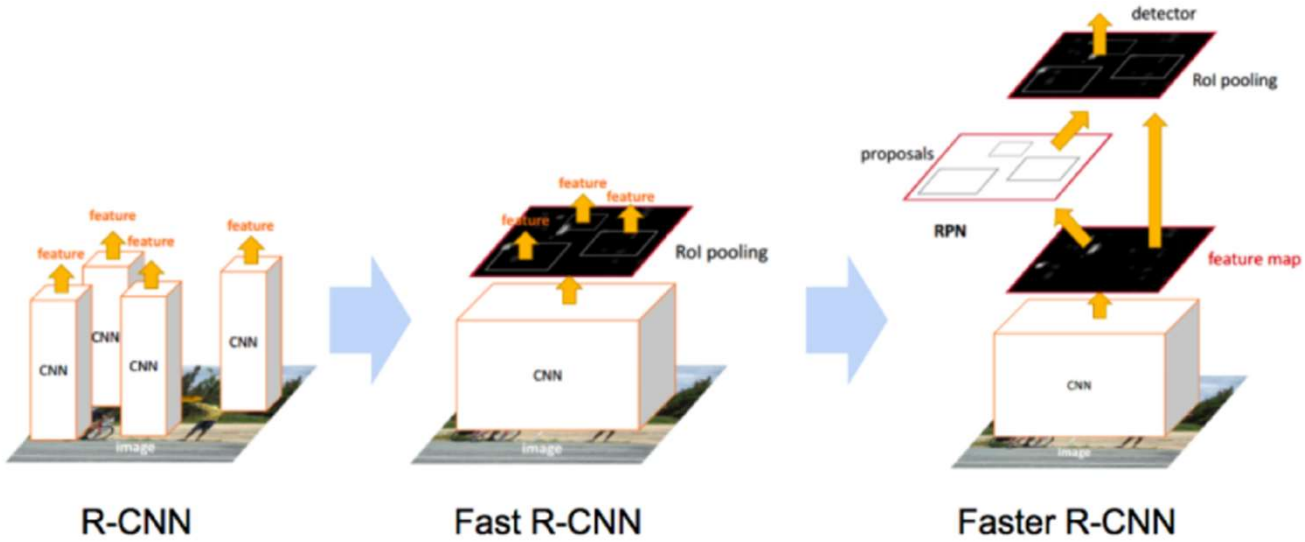
- Region Proposal을 RPN이라는 네트워크를 이용하여 수행(병목현상 해소)
- Region Proposal 단계에서의 bottleneck 현상 제거
    a. 해당 단계를 기존의 Selective search 가 아닌 CNN(RPN)으로 해결
- CNN을 통과한 Feature map에서 슬라이딩 윈도우를 이용해 각 지점(anchor)마다 가능한 바운딩 박스의 좌표와 그 점수를 계산
- 2:1, 1:1, 1:2의 종횡비(Aspect ratio)로 객체를 탐색

Faster R-CNN



1. RPN은 ImageNet을 사용하여 학습된 모델로부터 초기화되어 region proposal task를 위해 end to end로 학습됩니다.
2. 윗 단계에서 학습된 RPN을 사용하여 Fast R-CNN 모델의 학습을 진행합니다. (초기화는 ImageNet의 학습 모델로)
3. 초기화를 위의 네트워크를 사용하여 RPN을 학습하는데 공통된 Conv layer는 고정하고 RPN에만 연결된 층만 학습합니다.
4. 공유된 Conv layer를 고정시키고 Fast R-CNN의 학습을 진행합니다.

알고리즘 비교

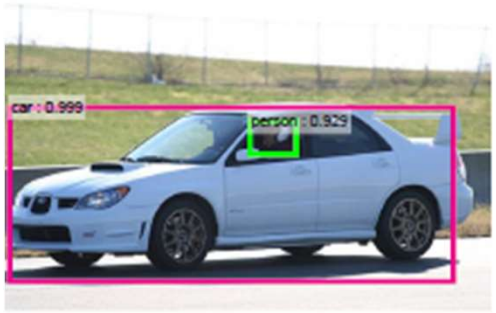


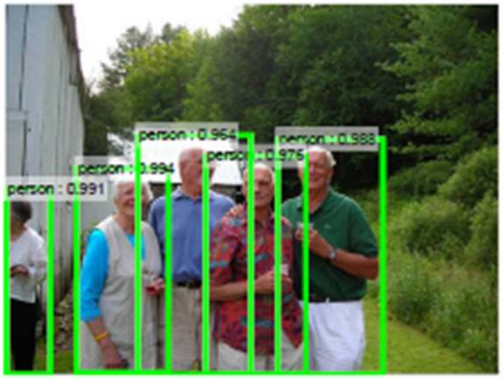
R-CNN Fast R-CNN Faster R-CNN

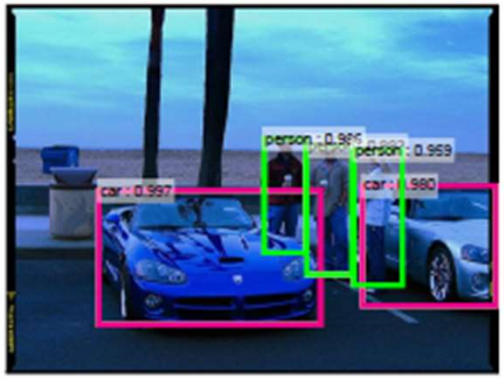
System	Time	07 data	07 + 12 data
R-CNN	~ 50s	66.0	-
Fast R-CNN	~ 2s	66.9	70.0
Faster R-CNN	**~ 198ms**	**69.9**	**73.2**

Detection mAP on PASCAL VOC 2007 and 2012, with VGG-16 pre-trained on ImageNet Dataset

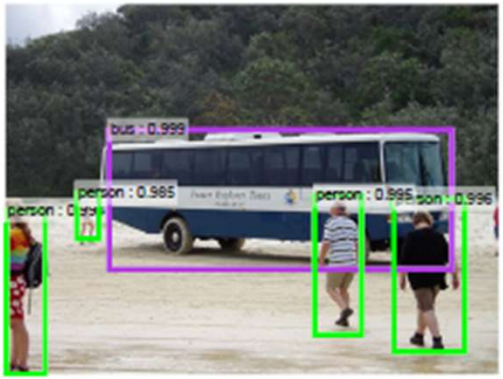
Result

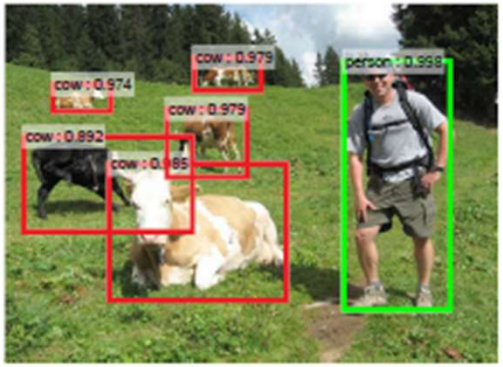

car : 0.999
person : 0.929


person : 0.964
person : 0.994
person : 0.991
person : 0.975
person : 0.988


person : 0.959
car : 0.997
car : 0.980


person : 0.961
person : 0.958
person : 0.960
person : 0.757
dog : 0.697


bus : 0.999
person : 0.985
person : 0.995
person : 0.996


cow : 0.979
cow : 0.974
cow : 0.979
cow : 0.892

참고

https://seongkyun.github.io/papers/2019/01/06/Object_detection/

https://woosikyang.github.io/fast-rcnn.html

https://woosikyang.github.io/faster-rcnn.html

감사합니다