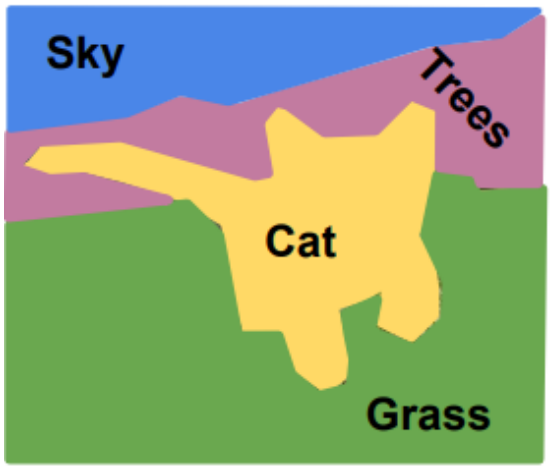
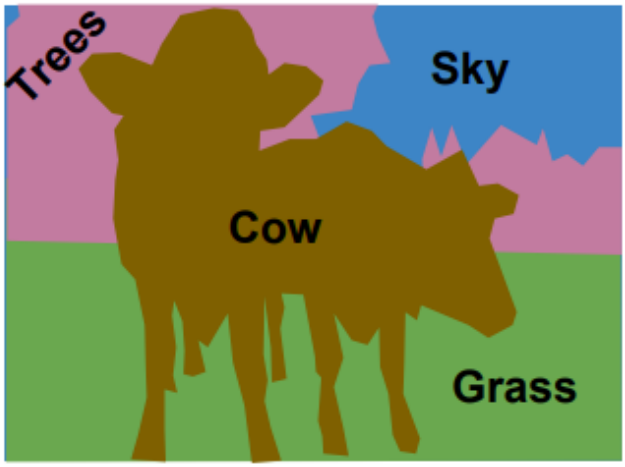
Lec 11. Detection and Segmentation

Semantic Segmentation

- input: image, output: category on each pixels
- 즉, 각 픽셀 별로 어떤 것을 나타내는 부분인지 결정
- classification처럼 카테고리 있음. but, 이미지 전체에 대한 카테고리가 아니라 픽셀 하나하나에 대한 카테고리
- 개별 객체를 구분하지 않음
    - ex) 소 두 마리가 있으면, 그 두 마리를 구분하지 못함. 그저 암소라는 픽셀 덩어리만 알 수 있음

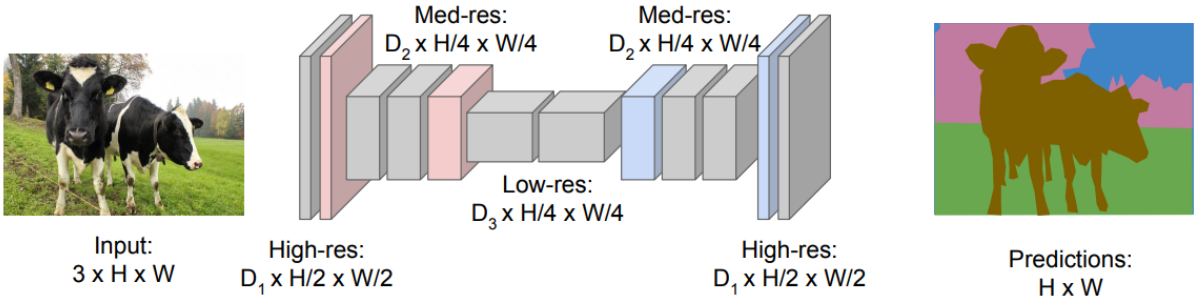




Classification을 통한 접근

- Sliding Window
  - input을 아주 작은 단위로 쪼개 그 작은 영역만을 가지고 classification 문제를 푼다고 생각
  - 비용이 많이 들어 좋은 방법은 아님 → 영역을 분할하는 경우에도 영역들끼리 공유하는 feature가 많을 것. 이를 개별적으로 적용하는 것은 비효율적
- Fully Convolutional Network
  - FC layer 없이 Convolution layer로만 구성된 network
  - output: C x H x W (C: 카테고리 수) → 모든 픽셀 값이 대해 classification score를 매긴 것
  - 모든 픽셀의 classification loss를 계산하고 평균값 취함 → back prop
  - input image의 spatial size를 계속 유지해야하기 때문에 비용이 큼
  - 그래서 feature map을 downsampling/unsampling하여 사용함

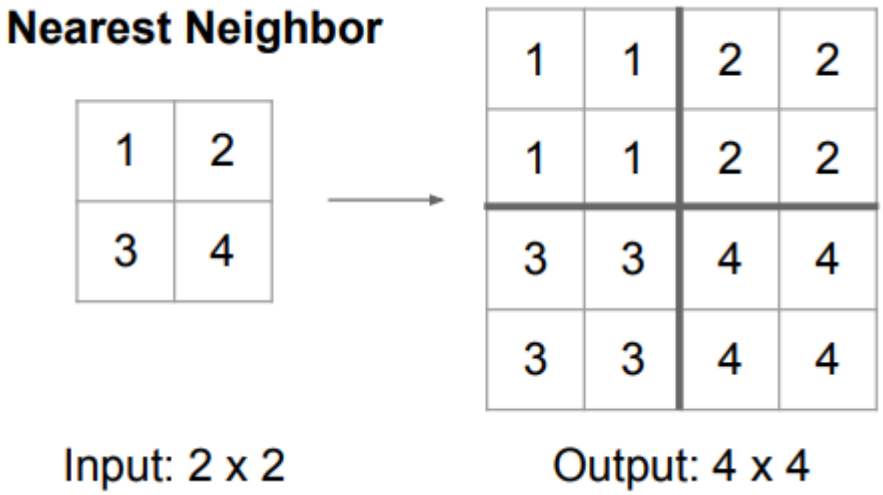
- classification과 유사. 차이점은 classsification에서는 FC layer가 있었으나, 이 network에서는 spatial resolution을 다시 키워 input image의 해상도와 같게 만드는 것 → 계산 효율이 좋음



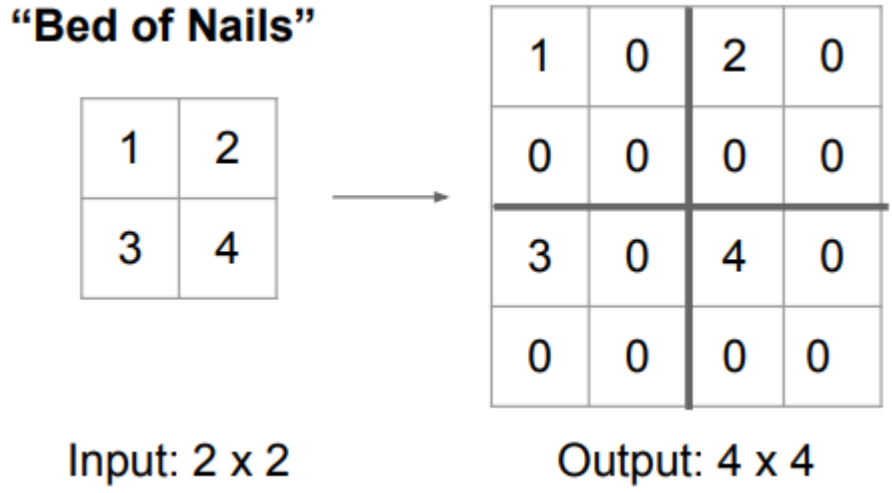
Unsampling

Unpooling(별도의 학습 x)

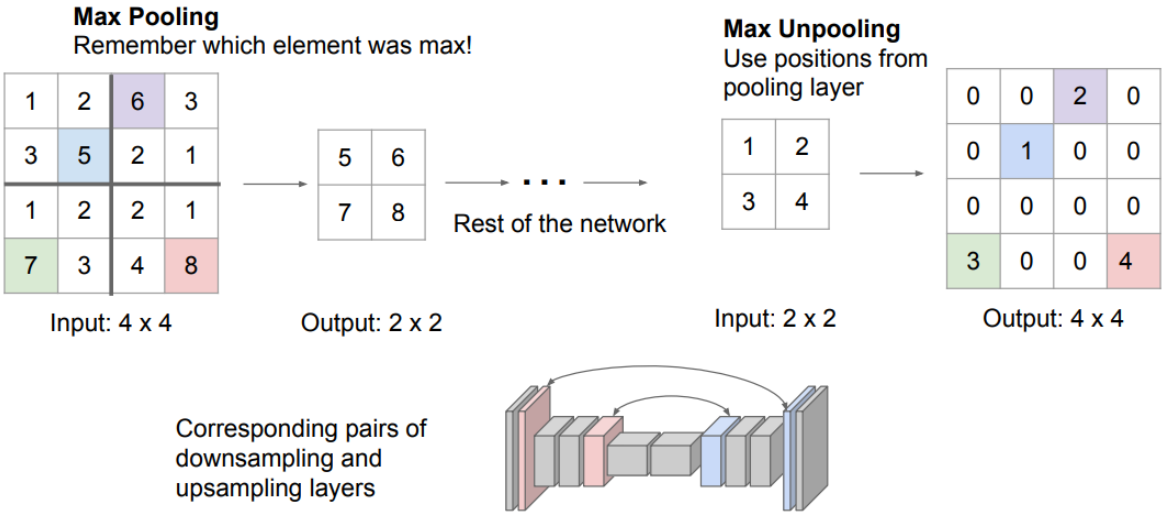
- nearest neighbor unpooling



- bed if nails unpooling/unsampling → zero region은 평평하고 non-zero region은 바늘처럼 값이 튐



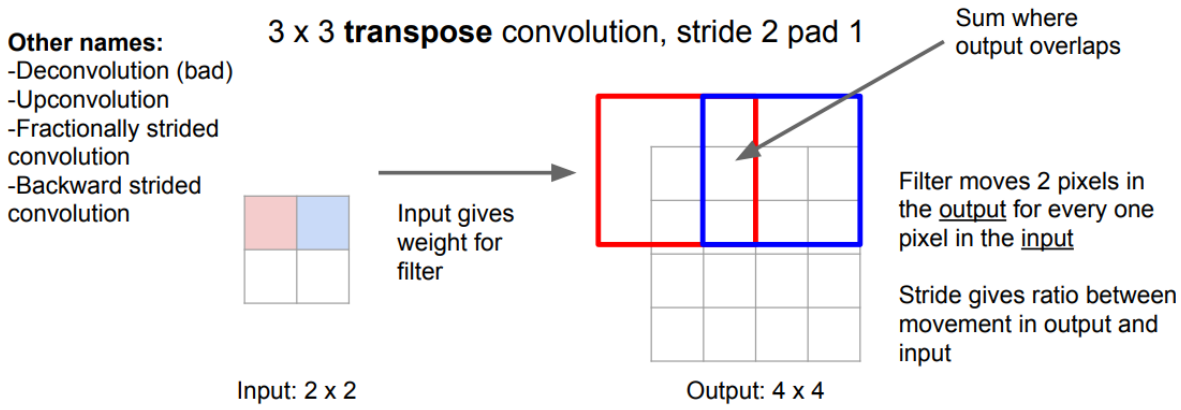
- max unpooling
  - 각 unpooling과 pooling을 연관지음
  - max pooling의 위치를 기억해뒀다가 이를 unpooling에서 기억하고 해당 위치에 값을 배치한 후 남은 부분에 0을 채워넣는 것
  - → low resolution feature map을 high resolution feature map으로, low resolution의 값을 maxpooling에서 선택된 위치에 넣는 것
  - 같은 위치에 넣어주니까 spatial information을 좀 더 디테일하게 다룰 수 있음 (max pooling에서 잃어버린 spatial info.를 유지하게 도와줌)



Transpose Convolution

- training 가능

- 특수한 방식의 convolution이라 할 수 있음
- 내적을 하지 않음
- feature map에서 하나의 값(input) 선택 → 그 값이 weight의 역할을하여 filter * input을 계산함 → input이 a만큼 움직일 때 output에서는 2a만큼 움직임 → 겹치는 부분이 있는 경우 두 값을 더해줌 → 반복



- transpose convolution은 행렬곱을 연산하되 그 행렬을 transpose시킴
- stride>2인 경우는 더이상 convolution이 아님 (normal convolution과 근본적으로 다른 연산이 됨 → transpose convolution이라는 이름이 붙게 된 이유)

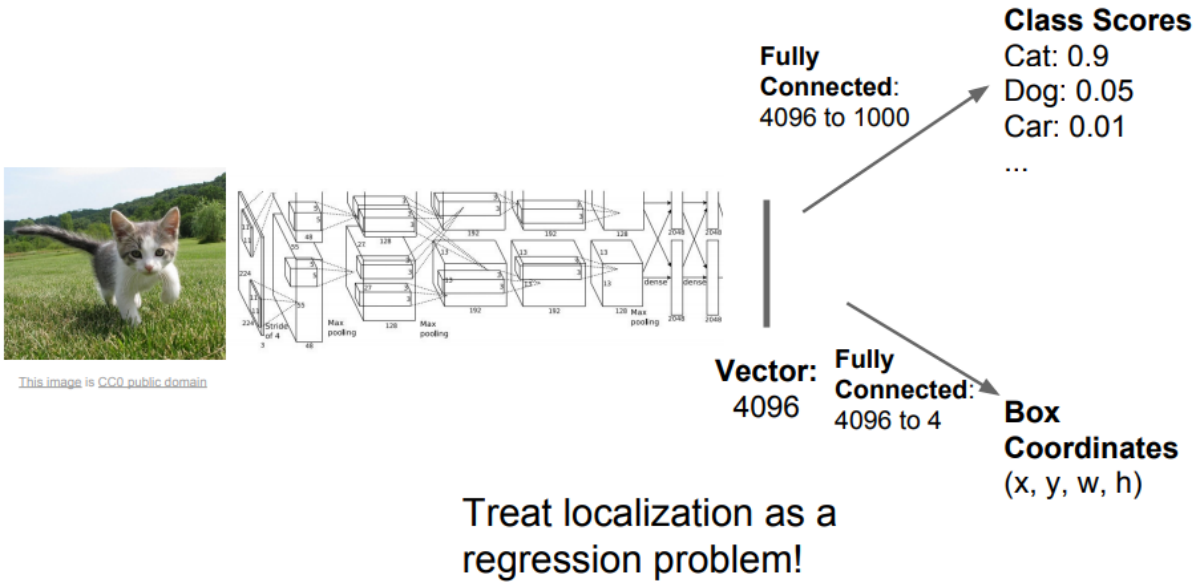
Classification + Localization

classification; input image의 label을 다는 문제

localization; 이미지 내에서 내가 관심있는 객체가 오직 하나 뿐이라고 가정

→ 어떤 이미지가 어디에 있는지를 확인하는 것. object detection과는 다름.

- image 내에서 객체 하나만을 찾아 labeling하고, 그 위치를 찾아내는 것
- 기본 구조



- output layer 직전의 FC layer는 class scores로 연결되어 카테고리를 결정함
- 또다른 fc layer는 4개의 원소를 가진 vector와 연결됨 → bounding box의 위치를 나타냄
- 즉, output은 class score와 객체의 위치
- loss
  - softmax: class score 예측을 위한 것
  - l2: ground truth bounding box와 예측한 bounding box사이의 차이를 측장하는 loss 중 가장 쉽게 디자인할 수 있는 것. l1 혹은 smooth l1을 사용해도 됨.
  - 다른 일반적인 방법은 loss값으로 비교하는 것이 아니라 다른 성능 지표를 비교하는 것. cross validation으로 하이퍼파라미터를 optimizing할 때 모델의 성능지표를 보는 것
- 예시) human pose estimation
  - input: 사람 이미지
  - output: 사람의 각 관절의 위치
  - 사람의 포즈를 예측함
  - 대부분 사람들의 관절 수는 같다는 가정 하에 네트워크 수행
  - 예측된 14개의 점에 대해 regression loss를 계산하고 backpropagation으로 학습시킴
  - L2 loss를 사용하거나 다른 regression losses를 사용할 수 있음

- regression loss: cross entropy나 softmax가 아닌 losses (continuous value에 관한 것)
- 관절의 위치는 continuous value이므로 regression loss 사용
- 추가적으로, continuous value를 어떠한 고정된 output으로 만들어내고자할 때 사용

Object Detection

- input image가 주어지면 해당 image에 나타나는 객체들의 bounding box와 category를 예측
- classification + localization과의 차이: 예측해야하는 bounding box의 수가 input image에 따라 달라짐 → 객체가 몇 개 있을지 모름
- object detection 문제는 localization과 꽤 다른데, detection 문제에서는 이미지마다 객체의 수가 달라지며, 이를 미리 알 수 없음 → regression 적용 어려움

Sliding Window (brute force)

- sematic segmentation에서 작은 영역으로 나누었던 것과 비슷한 방법 이용
- input image를 다수의 영역으로 나누어 처리
- 작은 영역에 대해 classification을 진행하며 해당 영역 내에 어떠한 객체가 있는지 확인함
- 이 때, background라는 카테고리를 추가해야함 → 어떤 카테고리에도 속하지 않을 때
- 너무 많은 경우의 수가 존재하며, 그 계산량은 매우 클 것

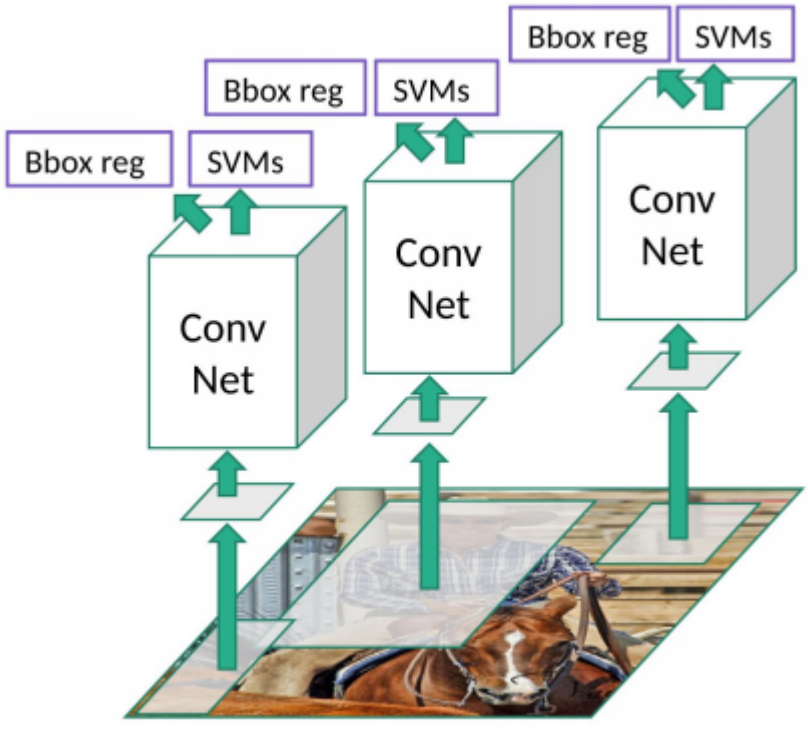
Region Proposal

- DL을 사용하지 않음 (다소 전통적인 방식)
- object가 있을법한 일정 개수의 bounding box 제공
- 이미지 내에서 묶여있는 것처럼 보이는(blobby) 곳을 찾아냄 → 객체가 있을 수 있는 후보 영역
- selective search
  - 2000개의 region proposal을 만들어냄
  - noise 심함
  - 대부분은 실제 객체는 아니지만 recall은 높음

- 이미지의 모든 부분에 대해 객체를 찾는 것이 아니라 region proposal network를 이용해 region proposal을 얻어낸 후 이들을 CNN의 input으로 넣는 것.
- 계산량을 다루기 쉬움

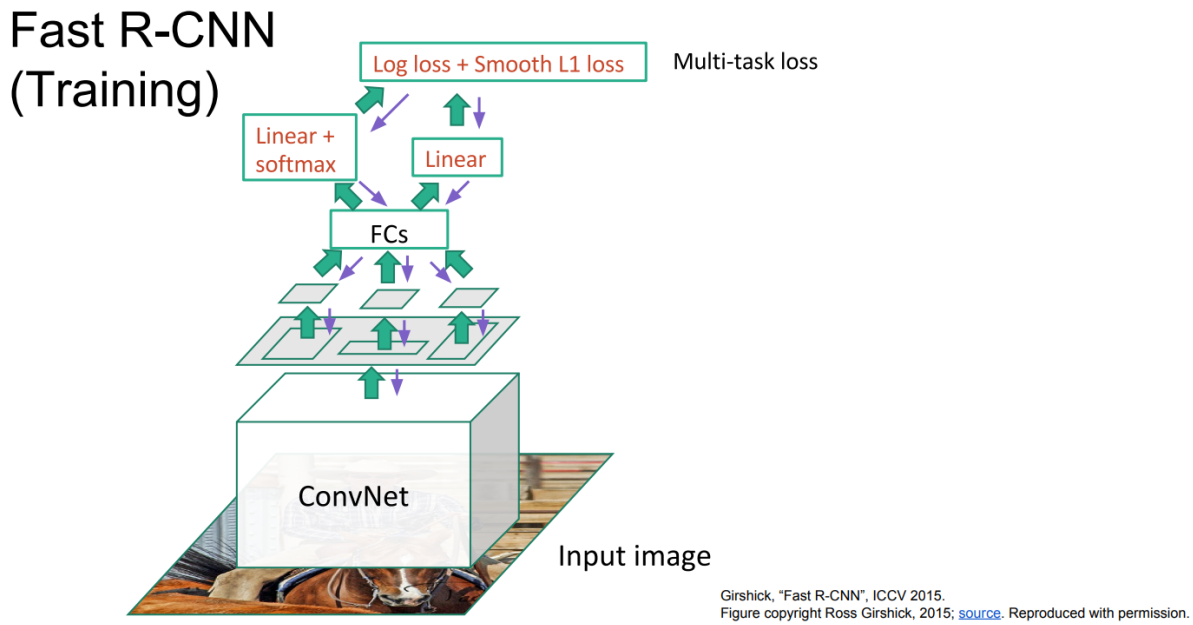
R-CNN

- selective search를 통해 2000개의 Region Proposal(Region of Interest/RoI)를 얻어냄
- RoI의 사이즈가 각각 다르다는 점이 문제가될 수 있음 → CNN의 input으로 이용하기 전 fixed size로 변형 필요
- fixed size의 input을 CNN에 통과시킴
- 마지막에 SVM 거침
- Region Proposals를 보정하기 위한 regression 과정 거침 (정확하지 못한 경우도 있기 때문)
- BBox category, offset(BBox 보정용) 예측 → Multi task loss를 두고 한 번에 학습
- R-CNN에서 region proposal은 주로 전통적 방식을 이용하므로 학습시키지 않음 (selective search)
- 🙁
    - high computation cost
    - 오래 걸림
    - 용량 큼



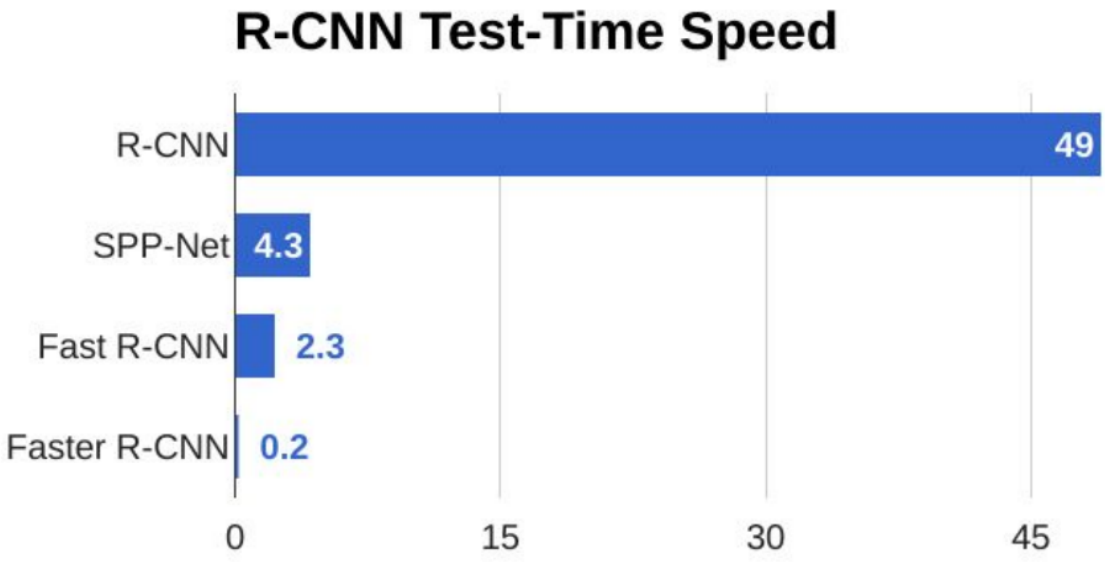
Fast R-CNN

- 이미지에서 RoI를 가져오지 않음 → 전체 이미지에 대한 CNN → 고해상도 feature map
- selective search같은 방법으로 region proposal을 계산함
- CNN feature map에 RoI를 projection, feature map에서 가져옴 → CNN의 feature를 여러 RoI가 공유 가능
- FC layer에 들어가기 전에 input의 size를 조정 → RoI pooling layer
- multi task loss



Faster R-CNN

- network가 region proposal을 직접 만들 수 있음
- 별도의 region proposal network가 있음
- RPN 이후의 나머지 동작은 fast R-CNN과 동일
- multi task loss 사용 (4개의 loss 한꺼번에 학습 → RPN loss(객체가 있는지 없는지, BBox에 관한 것), 최종단 loss(region proposal의 classification, region proposal 보정))



ETC

- YOLO: you only look once
- SSD: single shot detection
- 각 task를 따로 계산하지 말고 하나의 regression 문제로 풀어보자는 것
- single shot methods → 후보 base BBox와 GT object를 매칭시키는 방법
- 한 번의 forward pass만으로 끝냄
- region based methods는 single shot methods보다 정확도가 높지만 속도가 느림 ← single shot methods는 RoI 당 별도의 연산 x

Dense Captioning

- object detection + image captioning
- 각 region의 caption을 예측하는 문제
- Faster R-CNN과 방식 유사
- RNN language model 도입

Instance Segmentation

- input image가 주어지면 객체 별로 객체의 위치를 알아냄 ← object detection과 유사
- 객체 별 segmentation mask를 예측해야 함
- semantic segmentation + object detection
- Mask R-CNN
  - CNN, RPN, project, 각 bounding box마다 segmentation mask 예측
  - Faster R-CNN과 유사한 부분과 다른 부분으로 나뉨
    - 유사한 부분: region proposal의 category, bounding box regression 예측
    - 아닌 부분: semantic segmentation을 위한 mini network ← 각 pixel마다 객체인지 아닌지 분류

- pose estimation도 가능함 (→ loss와 layer 하나를 더 추가하면 됨)
- forward pass 한 번으로 수행 가능