exam 개인학습

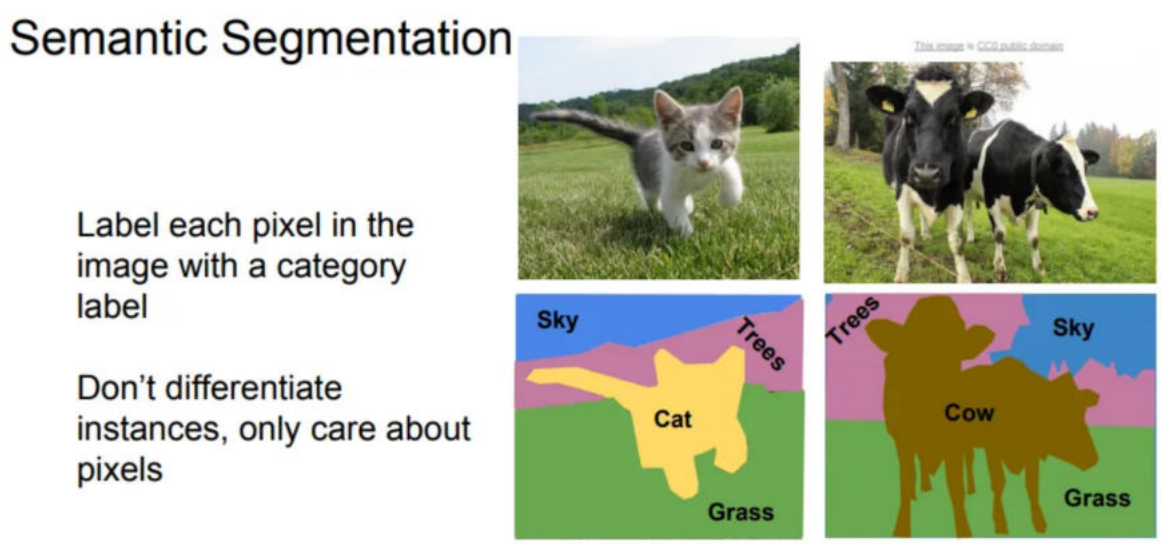
Lec 10

AI summary Try on this page

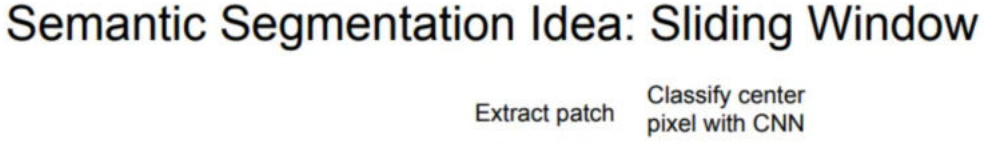
Type something...

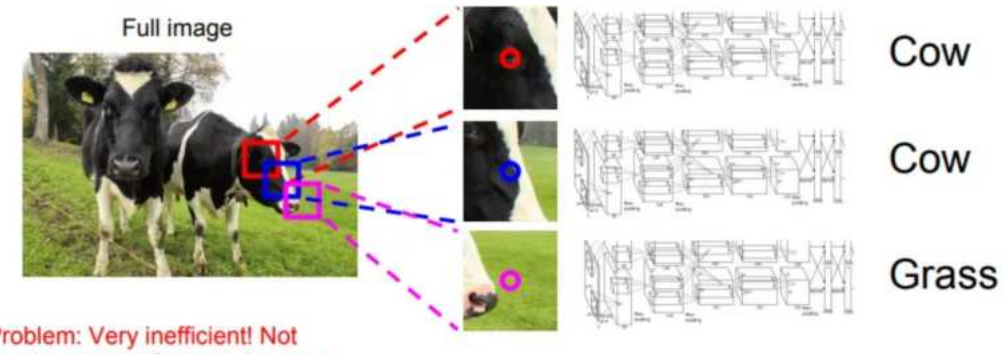
01) Detection and Segmentation

01_Semantic Segmentation



- 입력으로 이미지가 들어오면 출력으로 이미지의 픽셀의 카테고리를 정하게 된다. 일종의 classification 문제이다.
- 모든 픽셀에 카테고리가 매겨지기 때문에 이미지에 개체가 여러 개여도 하나로 인식되는 단점이 있다.

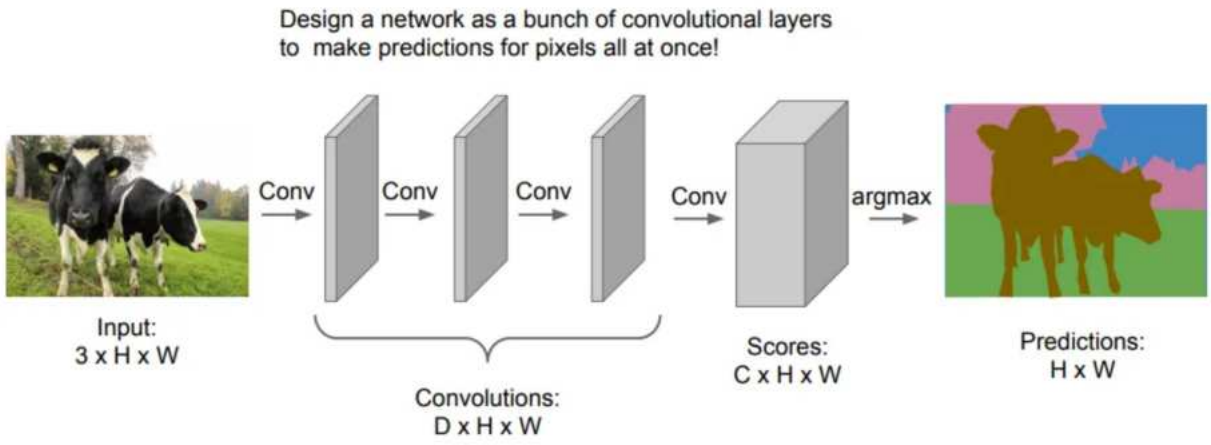




Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

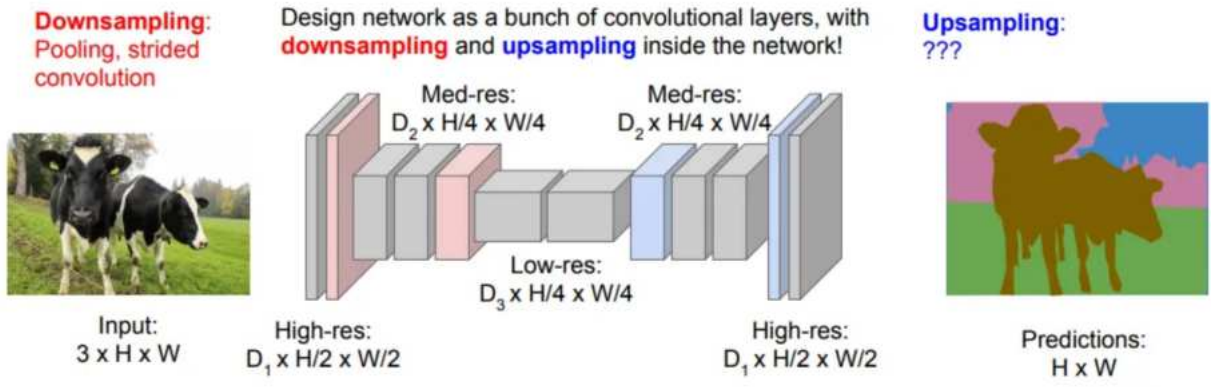
- 처음에 생각한 방식은 Sliding Window였다.
- patch를 만들어 이미지를 작은 단위로 쪼개고 각각의 patch마다 classification을 진행하는 방식이다.
- 하지만 비용이 크기 때문에 좋지 않은 방식이다.

Semantic Segmentation Idea: Fully Convolutional



- 2번째 고안한 방식은 Fully Convolutional 이다.
- FC layer가 존재하지 않고 Conv layer만 존재한다.
- 각각의 layer들이 spatial을 보존해준다.
- 하지만 이 방식 또한 계산 비용이 크다는 단점이 있다.

Semantic Segmentation Idea: Fully Convolutional



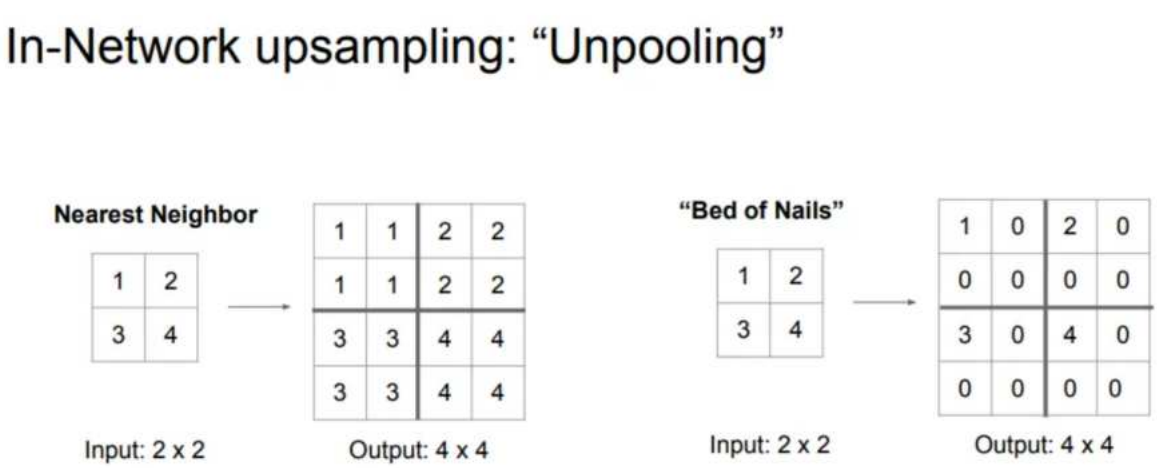
- 그래서 나온 방식이 Downsampling과 upsampling이다.
- 절반씩 줄이다가 다시 2배씩 늘리는 방식으로 사이즈를 보존한다.
- Downsampling은 pooling, stride를 이용해서 가능하다.
- 하지만 Upsampling은 바로 떠오르지 않는다.

1. Upsampling

- Upsampling은 커진 이미지의 빈 공간을 채우는 방식을 말한다.
- 강의에서는 3가지 방식을 제안한다

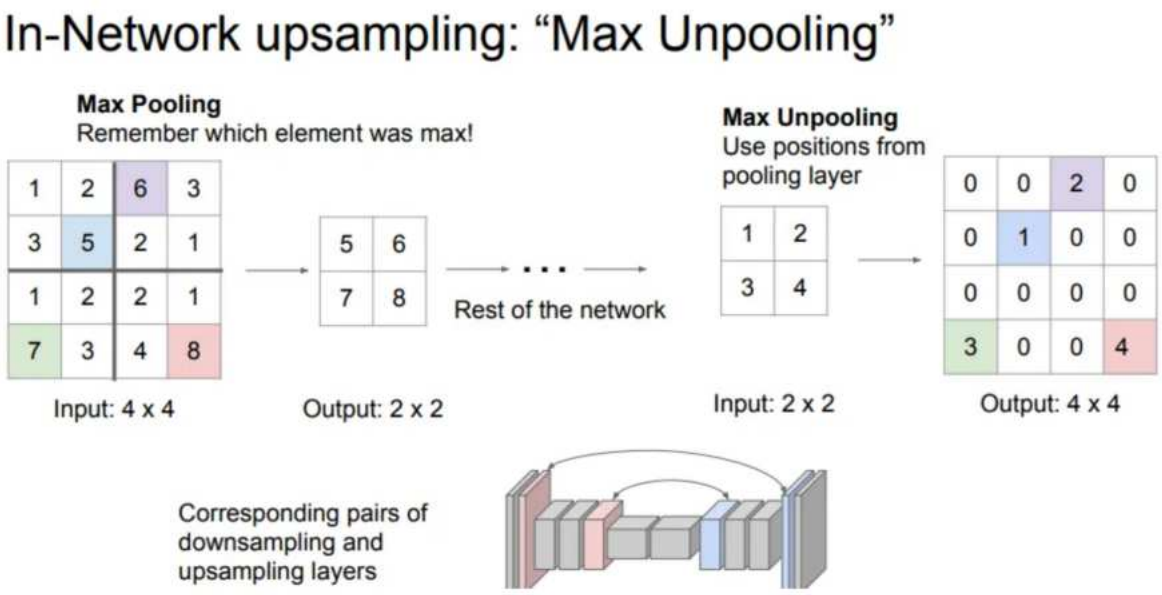
1. Unpooling
2. Max Unpooling
3. Transpose Convoluion

1.1 Unpooling



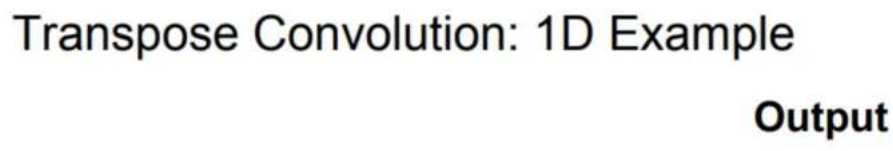
- Nearest Neightbor은 주변에 똑같은 값을 넣는 방식이고, Bed of Nails는 1번째 위치에 기존 input 값을 넣고 나머지를 0으로 채우는 방식이다.

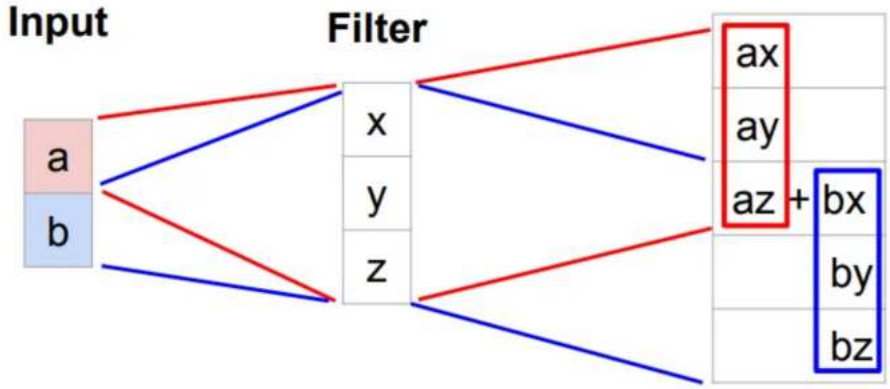
1.2 Max Unpooling



- Max pooling을 진행할 때 그 위치를 기억해서 다시 unpooling을 진행할 때 그 위치에 값을 집어넣는 방식이다.
- 공간 정보를 균형있게 유지할 수 있기 때문에 좋은 방식이다.

1.3 Transpose Convolution





- input을 필터에 곱하여 벡터를 만들어 낸다.
- 이때 겹치는 부분은 서로 더해준다.

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

- transpose한 convolution을 matrix multiplication 하는 과정으로 이해할 수 있다.
- 왼쪽은 convolution 을 matrix multiplication으로 나타낸 것이다.

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=2, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

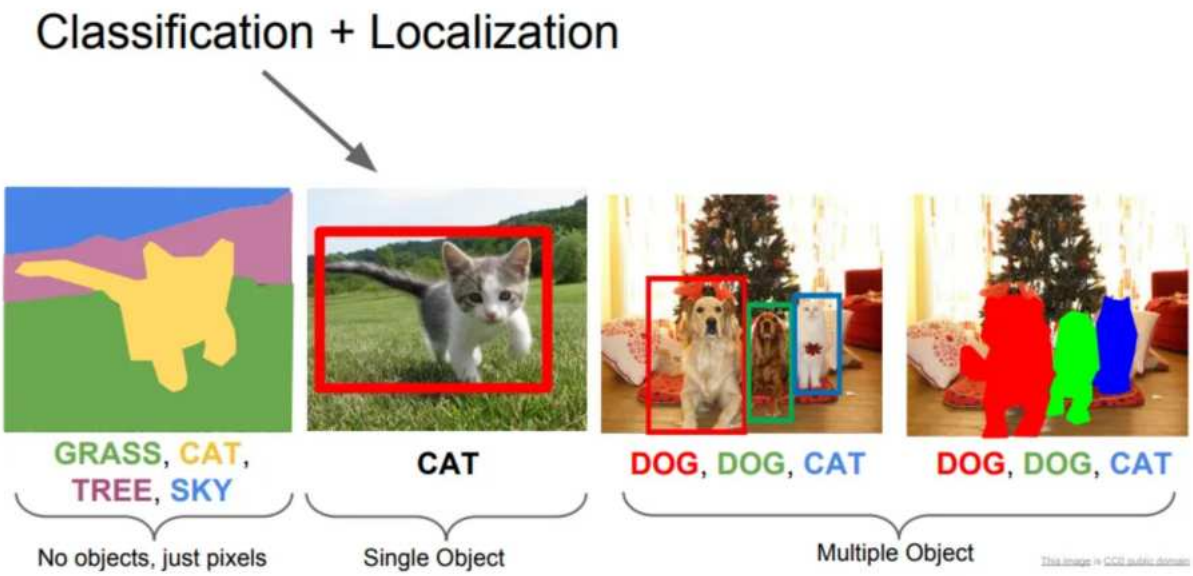
$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

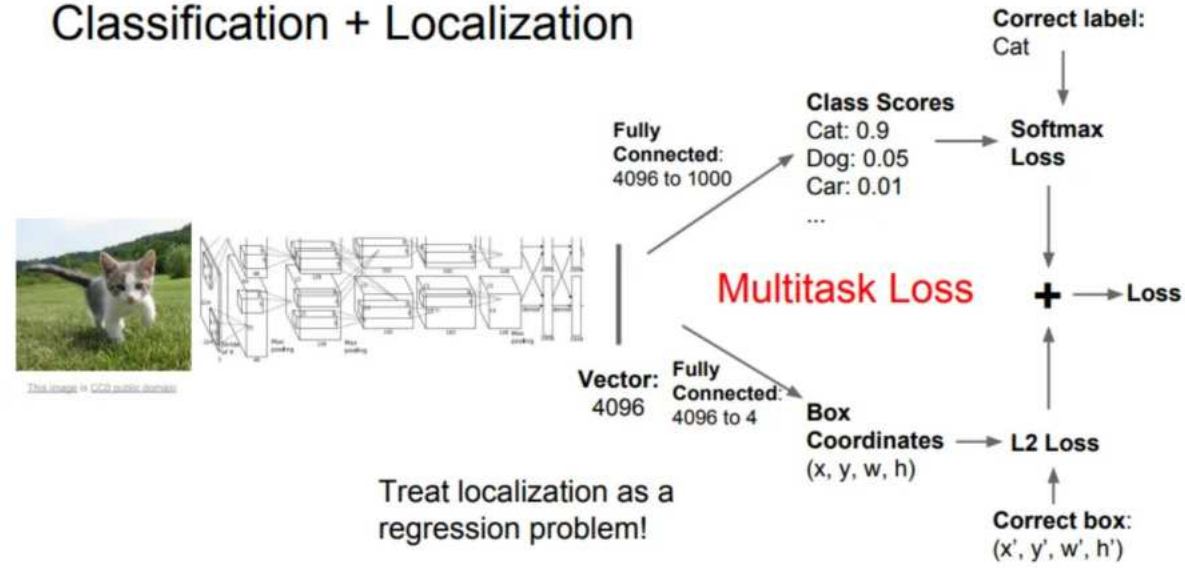
When stride>1, convolution transpose is no longer a normal convolution!

- stride가 2이면 convolution은 사이즈를 줄이지만 transpose는 줄이지 않고 펼치는 역할을 해준다.

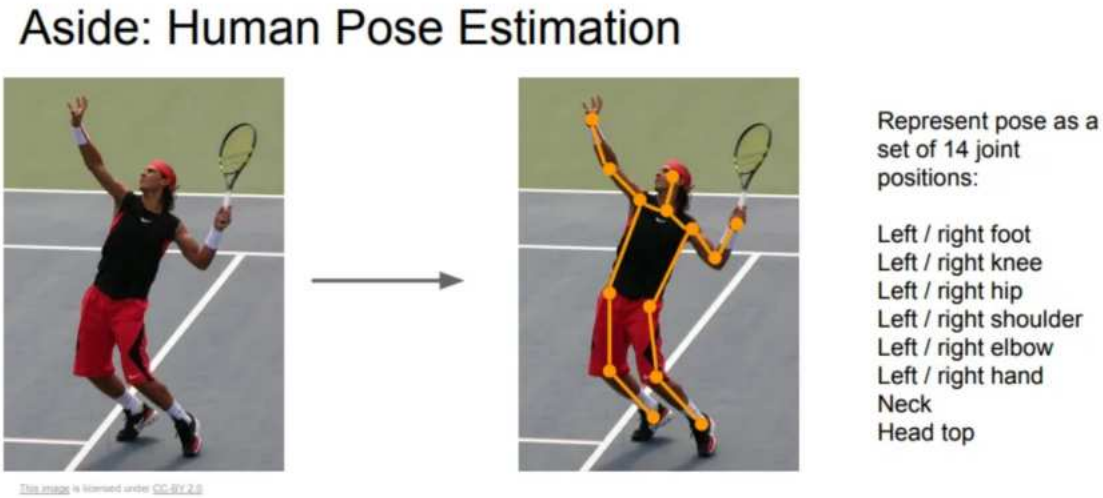
02_Classification + Localization



- classification에다가 object의 위치를 표시하는 네모박스를 그리는 것을 Localization이라고 한다.

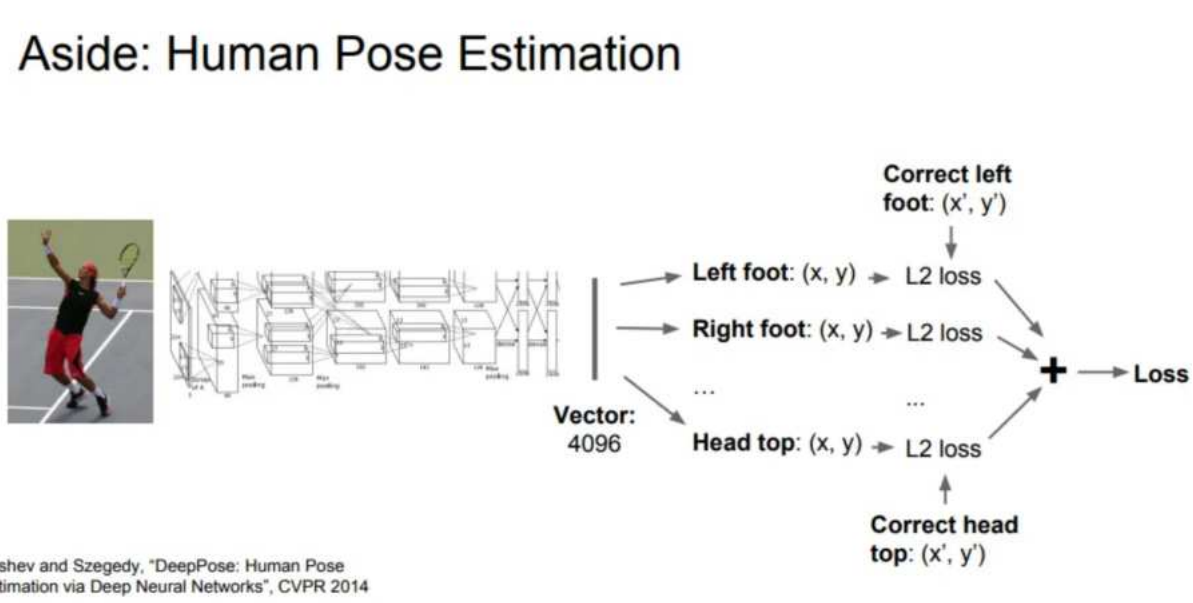


- score과 좌표를 output으로 만들어야 하므로 2개의 FC layer가 사용된다.
- 2개의 loss function을 이용해서 loss를 구한 후 더하여 loss값을 계산한다.
- 각각의 loss에는 가중치가 부과되서 loss로 계산된다.



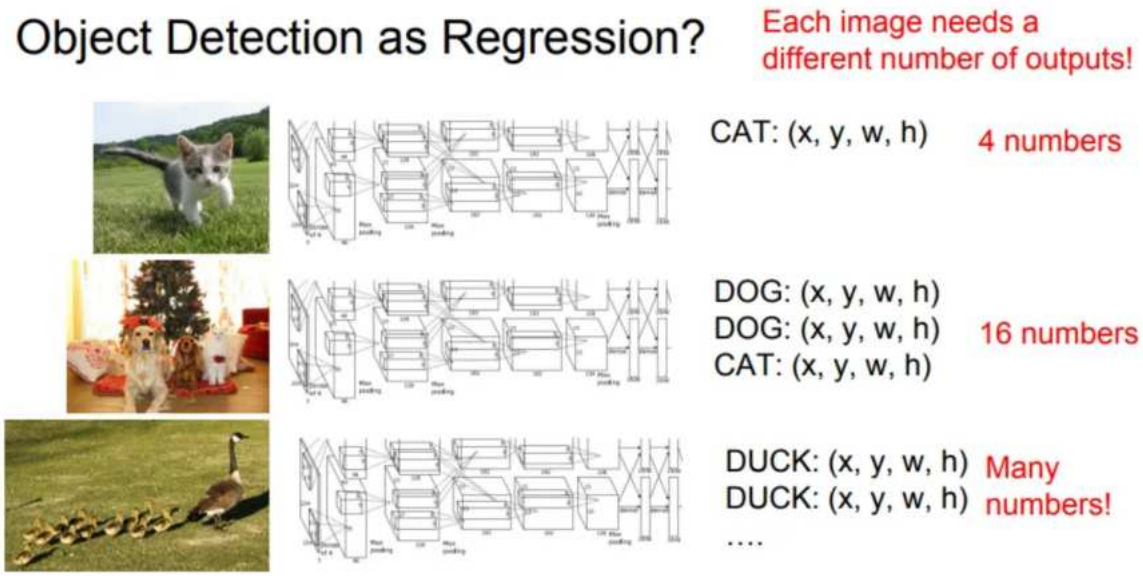
Johnson and Everingham, "Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation", BMVC 2010

- 인간의 joint부분도 localization을 통해서 구해낼 수 있다.



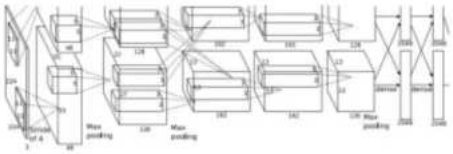
- 각각의 부분을 L2 loss로 구한 후 합하여 loss를 계산한다.

03_Object Detection



- Object Detection은 1장의 이미지에서 다수의 물체를 찾고, 그 물체가 어디있는지 알아내는 Task이다.
- Classification + Localization과 다른 점은 이미지 마다 **객체의 수가 달라져서 이를 미리 예측할 수 없다는 것이다.**

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations and scales, very computationally expensive!

- 매번 다른 crop을 만들어서 sliding 하는 방식의 brute force는 매우 비용이 비쌉니다.
- 따라서 object가 있는 곳을 먼저 proposal을 받는 알고리즘을 사용합니다.

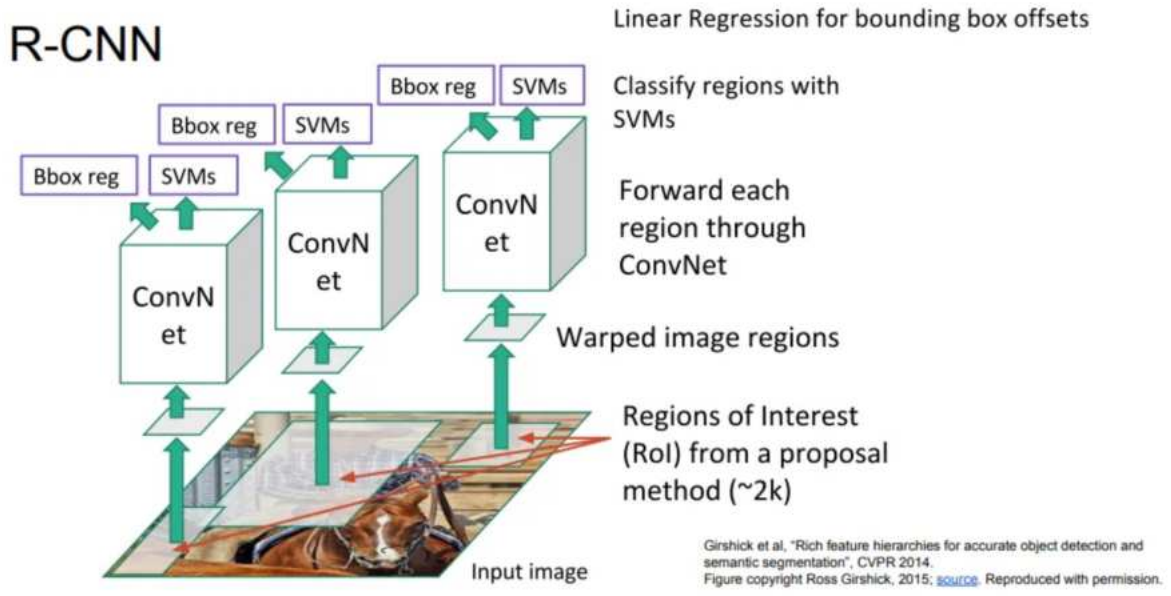
Region Proposals

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU

Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

- Region prooisal을 통해 object가 있는 위치를 찾아낸다.
- 후보로 뽑아낸 위치에서만 CNN을 사용한다.

1. R-CNN



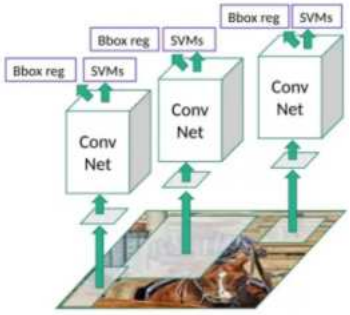
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

- Region of interes(ROI), 객체가 있을 법한 곳들을 2000개 찾는다.

- 각각의 크기가 다르기 때문에 크기를 변형시켜준다.
- CNN을 통과하면 최종분류에 SVM과 Bbox reg을 통해 좌표도 output으로 만든다.

R-CNN: Problems

- Ad hoc training objectives
  - Fine-tune network with softmax classifier (log loss)
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (least squares)
- Training is slow (84h), takes a lot of disk space
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman. ICLR15]
  - Fixed by SPP-net [He et al. ECCV14]

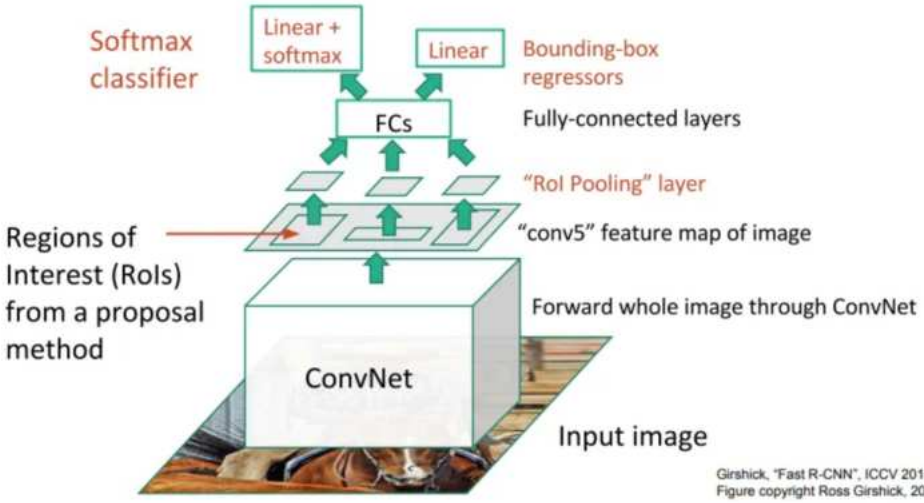


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Slide copyright Ross Girshick, 2015; source. Reproduced with permission.

- 비용이 비싸고, 훈련이 느리며 region proposal에 굉장한 시간을 쏟는다는 단점이 있다.

2. Fast R-CNN

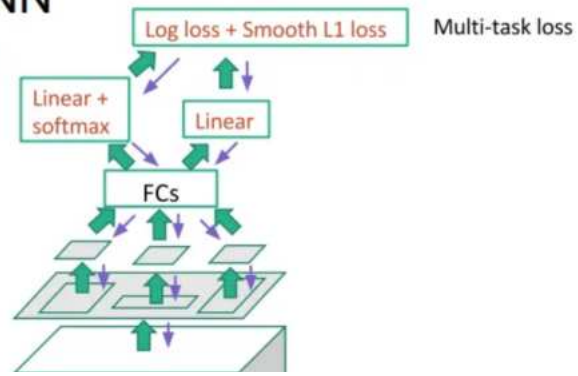
Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

- Region을 나누는고 분류하는게 연산적으로 너무 비싸다는 단점을 해결하기 위해 Fast R-CNN이 등장하였다.
- 이미지를 ConvNet에 넣어서 나온 결과인 Feature map에 대하여 Reigion of Interest를 구하고 `pooling` 을 하여 고정된 크기의 입력(사이즈가 다른 것을 맞춰줌)으로 변경한 후 통일된 Fully-Connected layer로 분류를 한 모델이다.

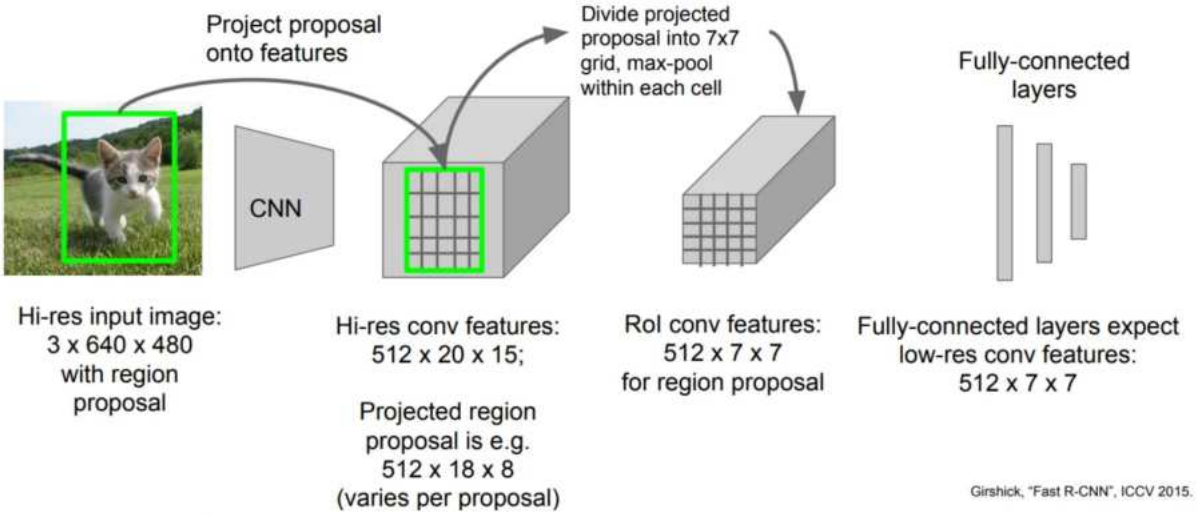
Fast R-CNN (Training)



Girshick, "Fast R-CNN", ICCV 2015.
Figure copyright Ross Girshick, 2015; source. Reproduced with permission.

- 학습할 때는 multi-task loss부터 backpropagation을 진행하여 학습시켰다.

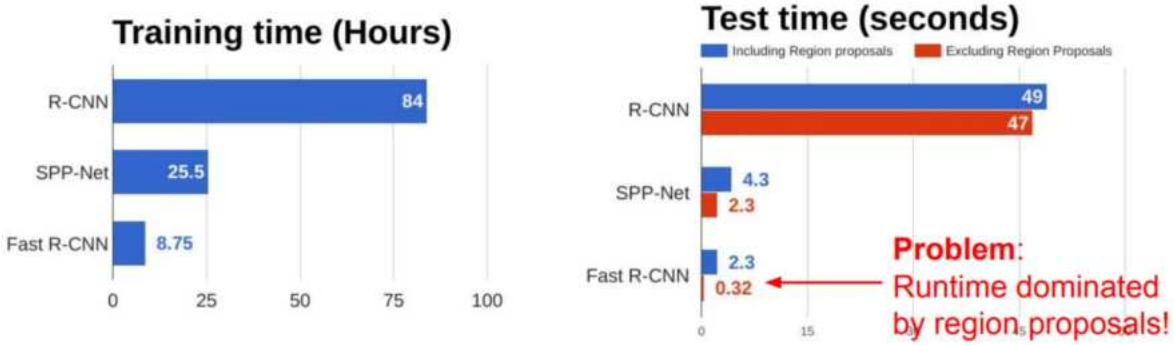
Faster R-CNN: RoI Pooling



Girshick, "Fast R-CNN", ICCV 2015.

- max pooling을 사용하여 특성 사이즈로 맞춰주었다.

R-CNN vs SPP vs Fast R-CNN

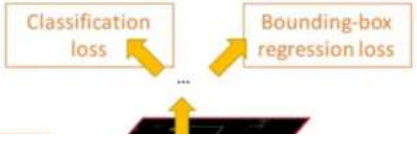


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
He et al, "Spatial pyramid pooling in deep convolutional networks for visual recognition", ECCV 2014
Girshick, "Fast R-CNN", ICCV 2015

- **fixed way region proposal algorithm**을 사용하기 때문에 Fast R-CNN의 Test time을 보시면, region proposal이 점유하는 시간이 2sec이다.

3. Faster R-CNN

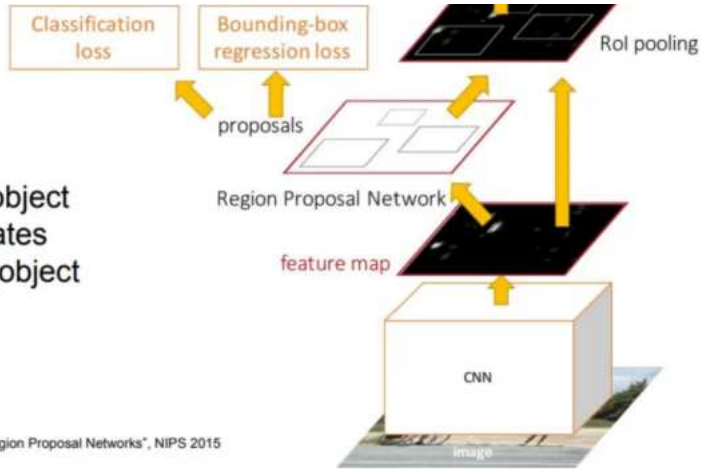
Faster R-CNN:
Make CNN do proposals!



Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



ten et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
igure copyright 2015, Ross Girshick; reproduced with permission

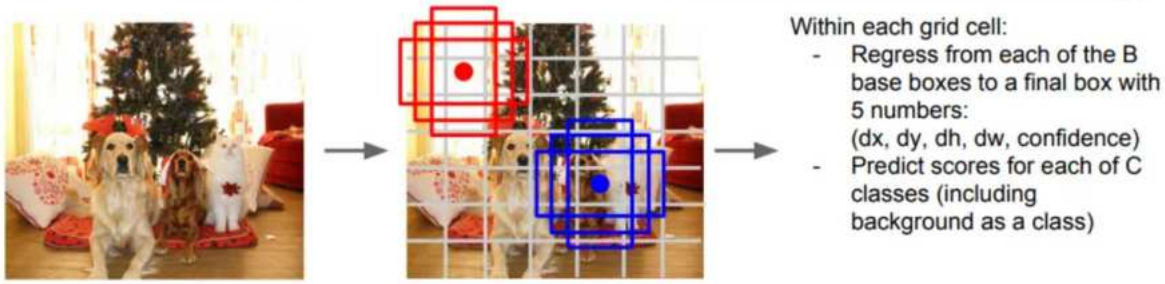
- 기존 Fast R-CNN이 느렸던 이유는 Region을 고정적인 갯수를 구하고 이 Region에 대해 각각 FC를 거쳐 객체를 분류했기 때문이다.
- 이를 해결하기 위해 하나의 이미지 입력으로 다수의 ROI(Region of Interest)를 출력하는 Region Proposal Network를 구성하고, 이렇게 나온 ROI들에 대해 Fast R-CNN를 거치는 방법이다.
- Loss는 총 4가지를 사용한다.

1. RPN이 object/not object인지 분류
2. RPN의 box 좌표 regress
3. Final 분류 점수
4. Final box 좌표

4. Detection without Proposal

Detection without Proposals: YOLO / SSD

Go from input image to tensor of scores with one big convolutional network!



Input image
3 x H x W

Divide image into grid
7 x 7

Image a set of **base boxes** centered at each grid cell
Here B = 3

Within each grid cell:

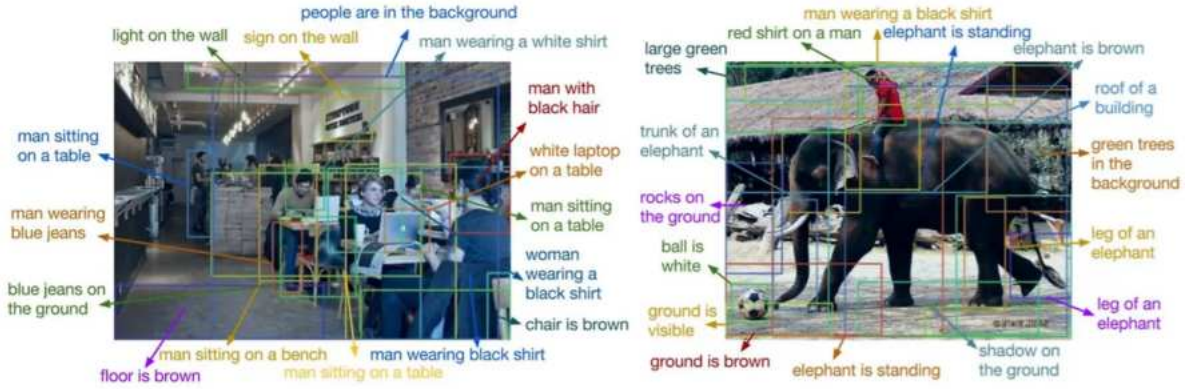
- Regress from each of the B base boxes to a final box with 5 numbers: (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

Output:
7 x 7 x (5 * B + C)

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

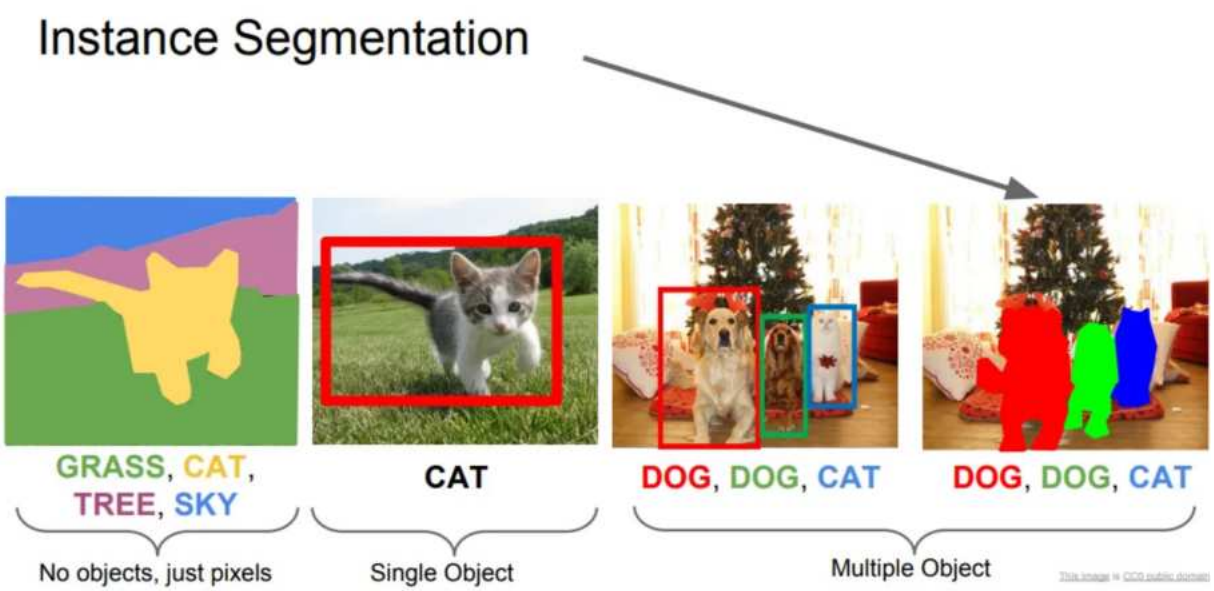
- 이미지에 대해 grid를 나누고 각 grid에 대해 dx, dy, dh, dw, confidence를 학습하는 방식이다.
- 박스 개수는 B개로 각 grid마다 B개의 base box를 사용한다.

Aside: Object Detection + Captioning = Dense Captioning



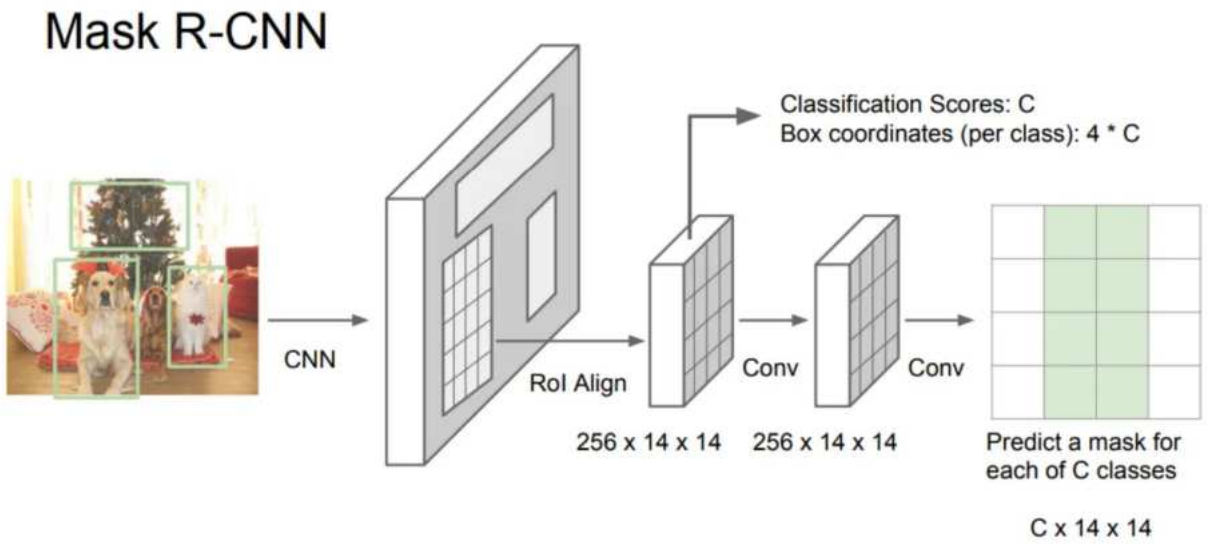
- object Detection과 Captioning을 합친 Dense Captioning도 있다.

04_Instance Segmentation



- Object Detection에서는 object의 위치를 박스로 표현했지만 Instance Segmentation은 각각의 object에 대해 semantic segmentation을 하는 것이다.

1. Mask R-CNN



He et al, "Mask R-CNN", arXiv 2017

- mask R-CNN이 Instance Segmentation을 잘한다고 알려져있다.
- 자세한 모델을 나중에 논문을 통해서 정리하겠다.