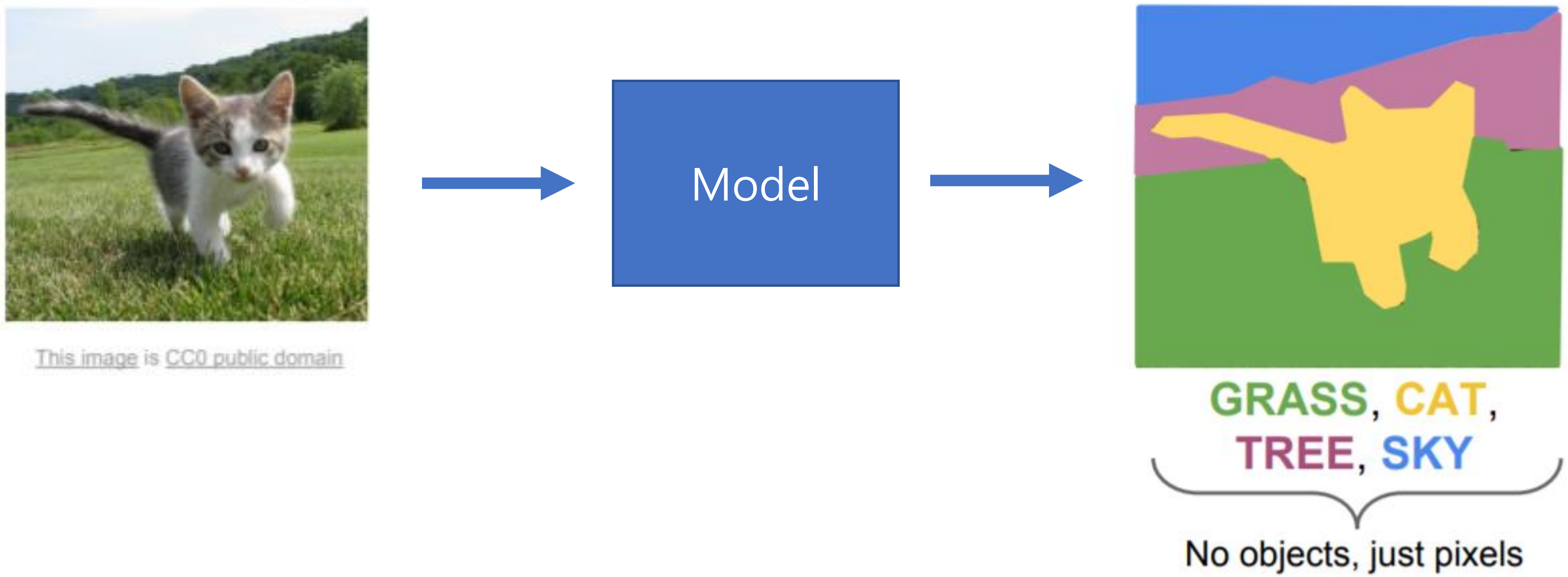
cs231n

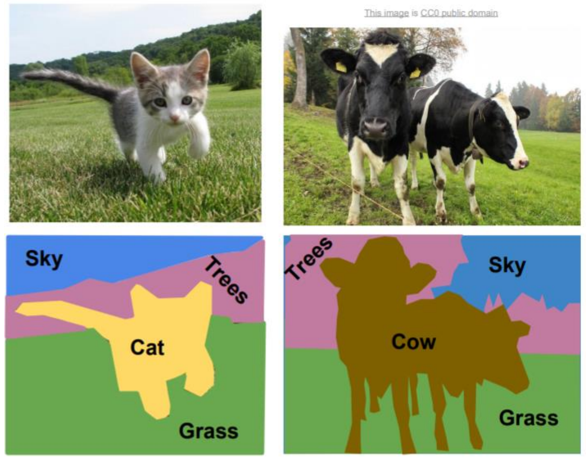
Lec 11,12

Lecture 11-computer vision Tasks

- Semantic Segmentation
- Classification + Localization
- Object Detection
- Instance Segmentation

Semantic Segmentation

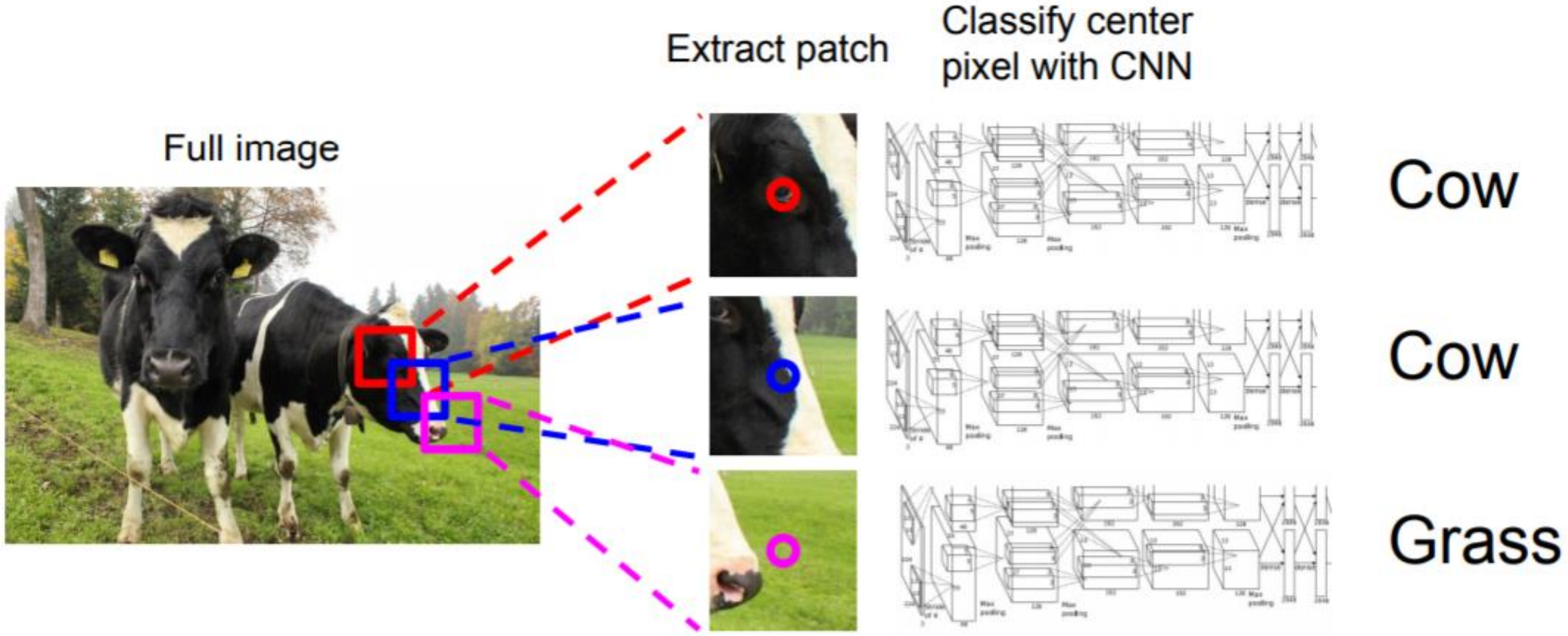




- Pixels – label
- Instance 구분 X, 즉 같은 암소는 같은 라벨

Semantic Segmentation

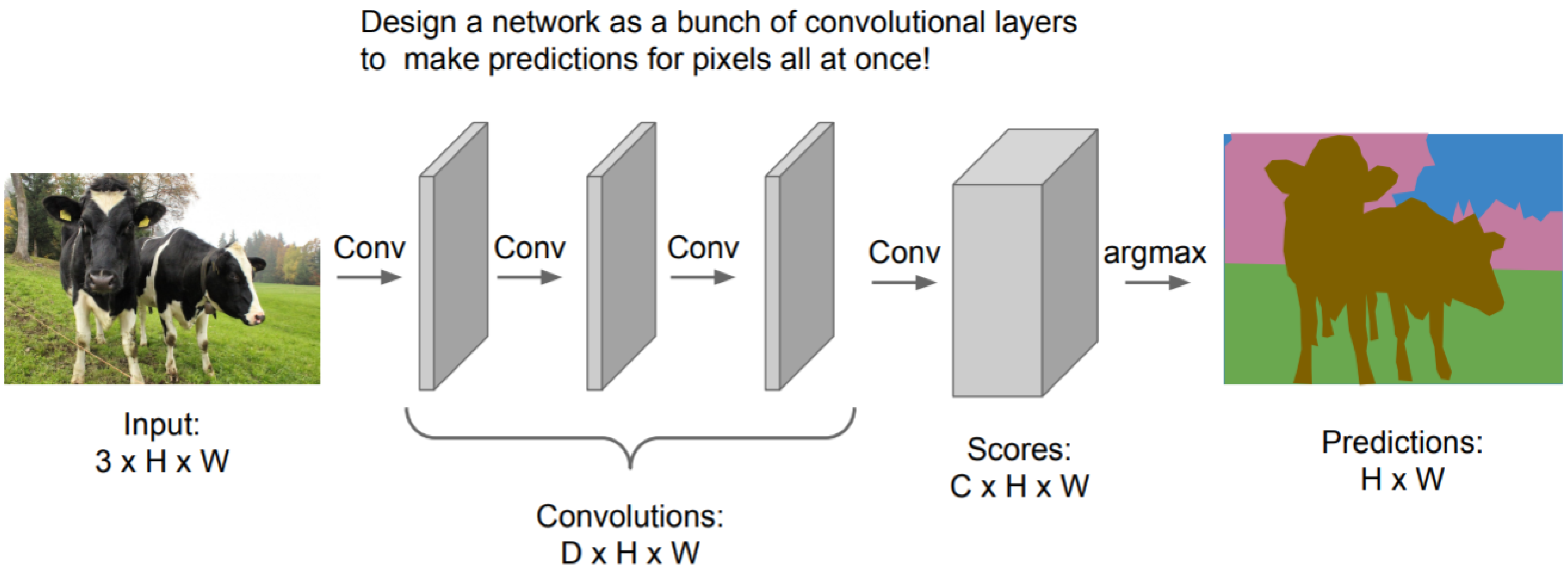
: Naive Idea – sliding window



개별 window 마다 모델을 학습.
→ 굉장히 비효율적이고 계산비용이 막대함.

Semantic Segmentation

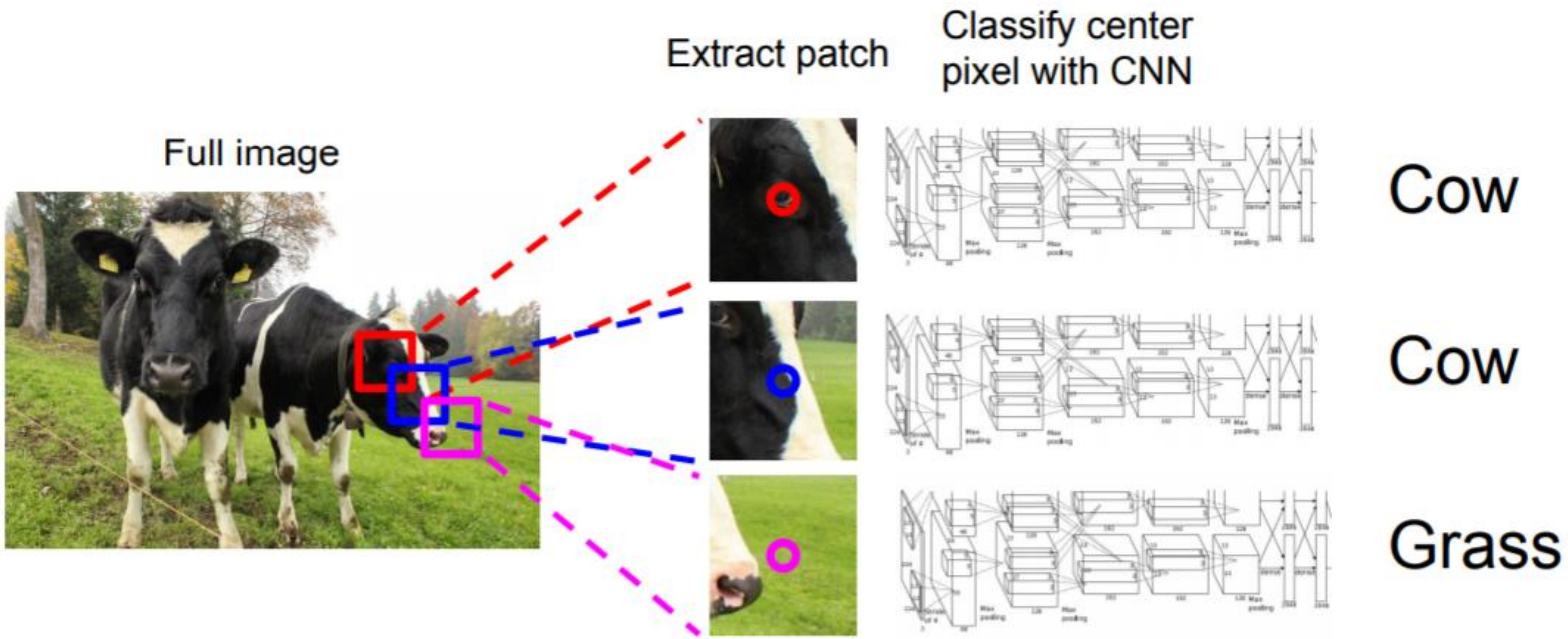
: Idea – FC layer



영역을 나누지 않고 모든 픽셀에 cross-entropy를 적용.
→ Spatial info를 계속 유지해야해서 여전히 계산비용이 비쌈

Semantic Segmentation

: Idea – FC layer

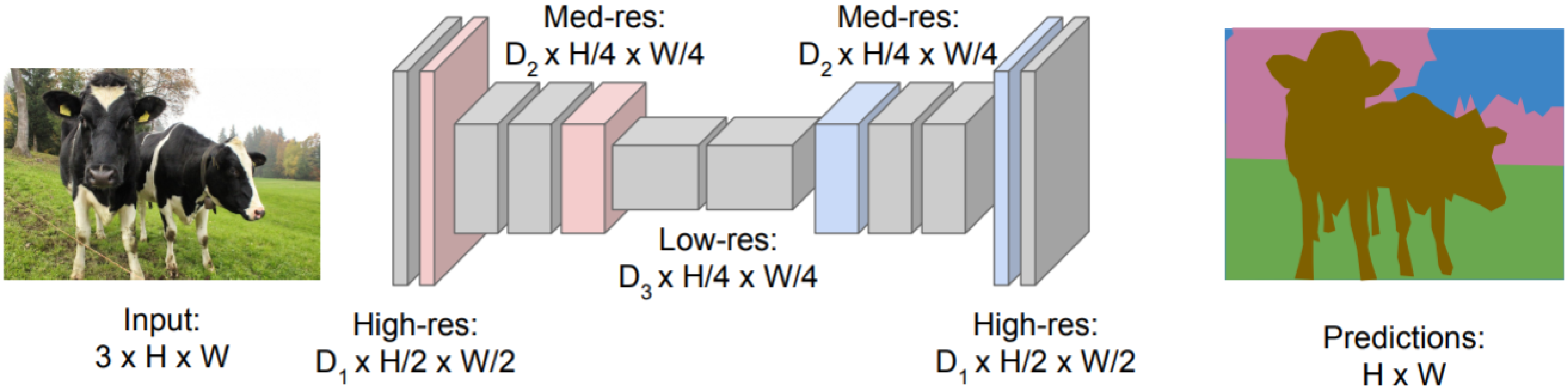


개별 window 마다 모델을 학습.
→ 굉장히 비효율적이고 계산비용이 막대함.

Semantic Segmentation

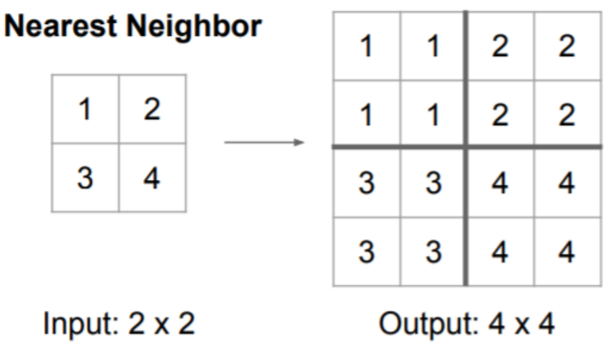
: Idea – FC layer

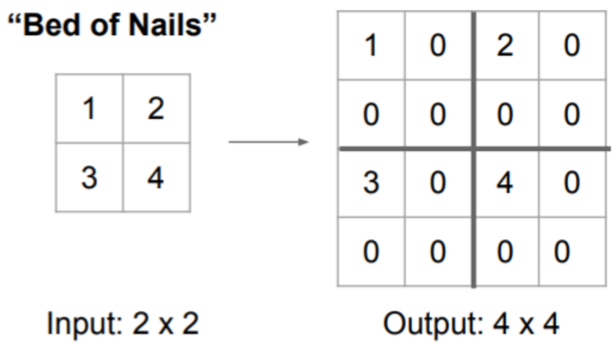
Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!



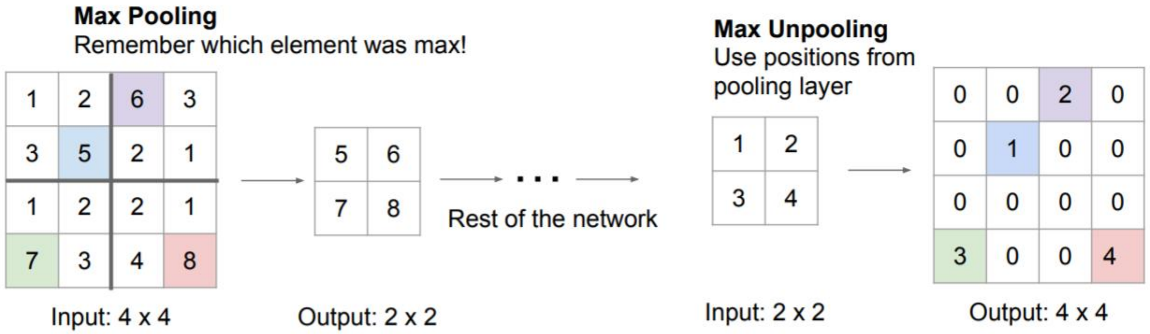
Upsamplinsg Downsampling

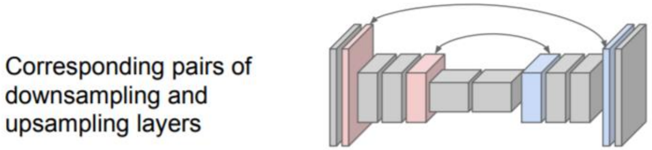
- Down Sampling – pooling, stride conv…
- Upsampling





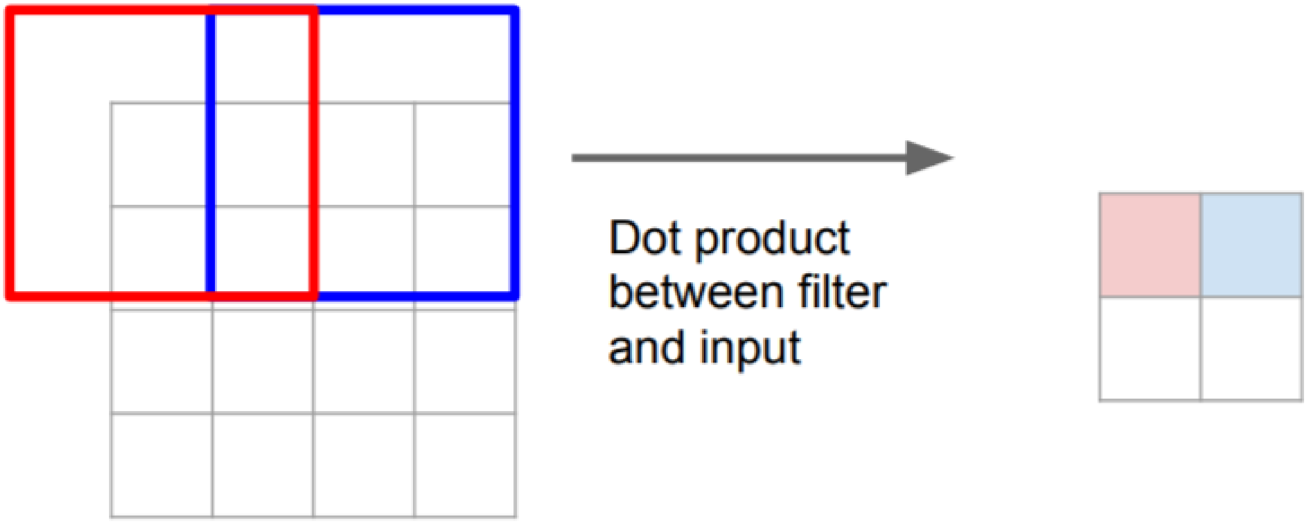
Spatial info 유지를 위한
대칭성 유지





Learnable Upsamplaing: Transpose Conv

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

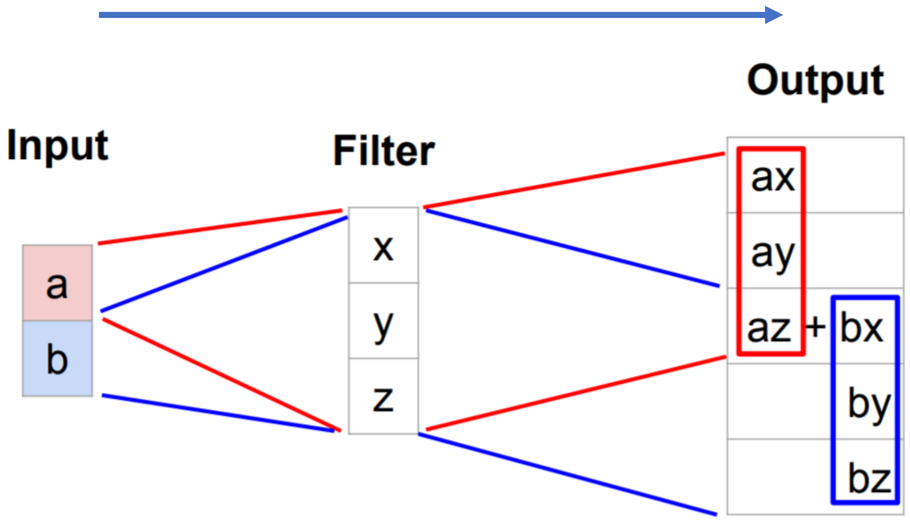
Output: 2 x 2

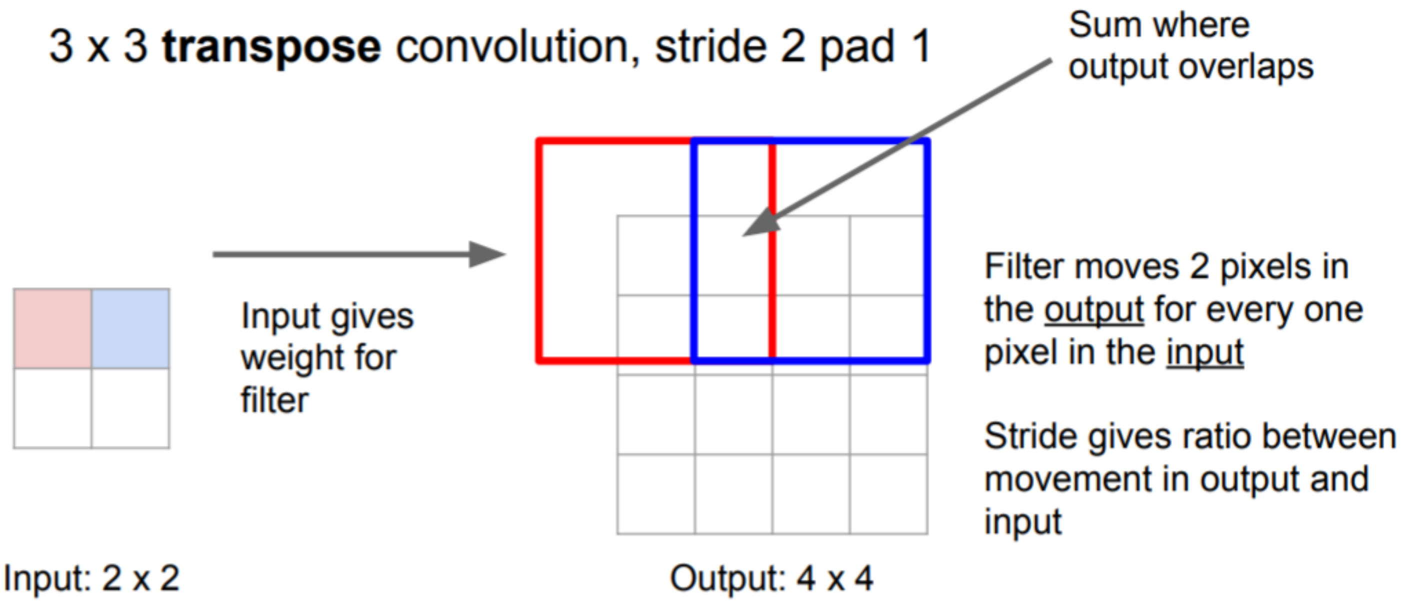
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Learnable Upsamplaing: Transpose Conv

- Deconv
- Upconv
- Backward strided conv
- Fractionally strided conv





Learnable Upsamplaing: Transpose Conv

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)

← Stride 1

Stride2 →

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:

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

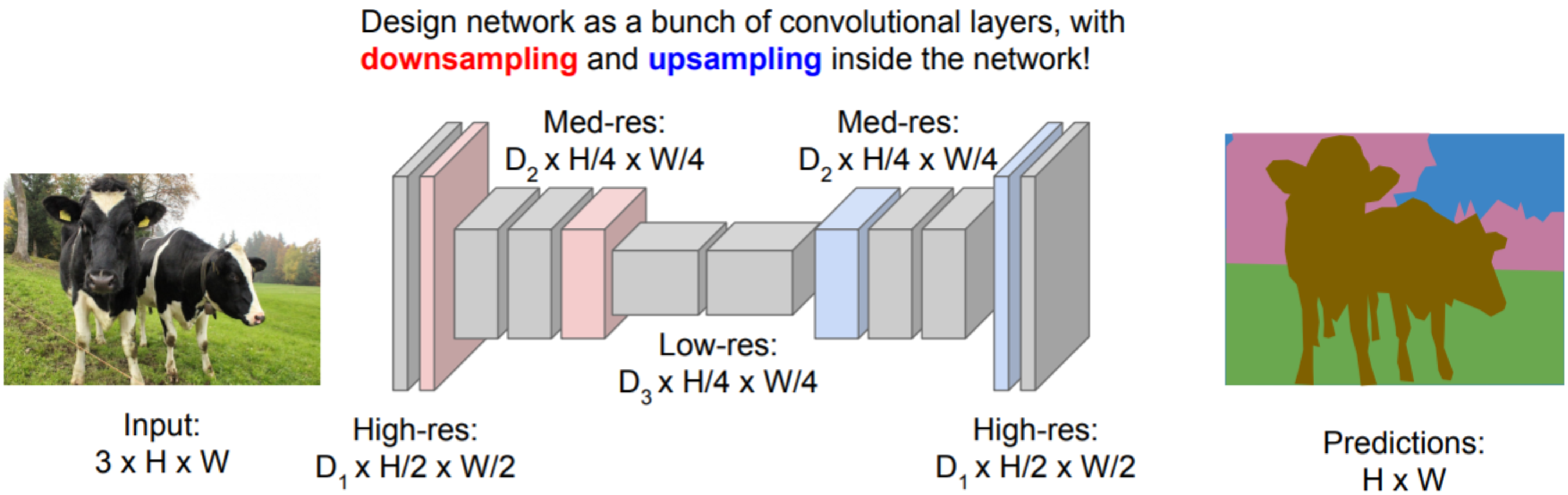
$$\begin{bmatrix} x & 0 \\ y & 0 \\ z & x \\ 0 & y \\ 0 & z \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ax \\ ay \\ az + bx \\ by \\ bz \\ 0 \end{bmatrix}$$

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

X,y,z are convolutional filter
Vector a is input

Semantic Segmentation

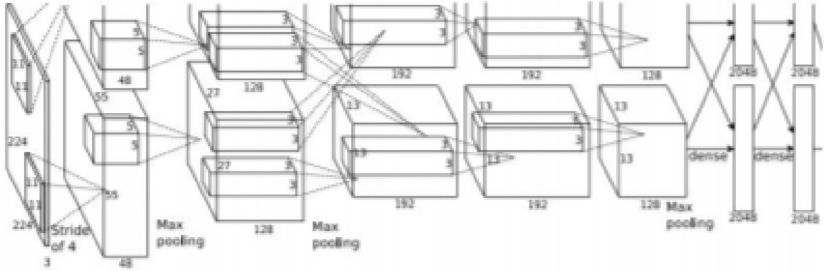
: Idea – FC layer



For every pixel, cross entropy를 계산하면 네트워크 전체를 end to end 로 학습.

Classification + Localization

↓pretrained된 모델을 사용
(transfer learning)



Correct label:
Cat

Fully
Connected:
4096 to 1000

Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Softmax
Loss

Multitask Loss

\+ → Loss

성능의 지표로는
조금 무리.

Vector:
4096

Fully
Connected:
4096 to 4

Box
Coordinates
(x, y, w, h)

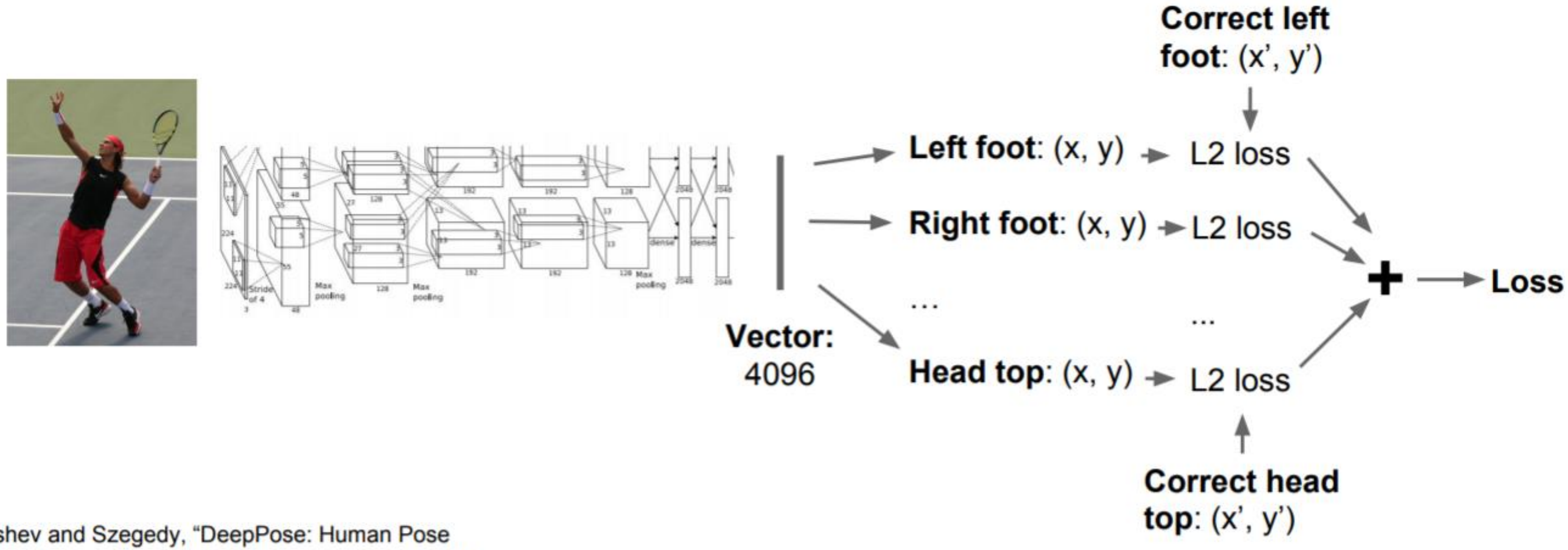
L2 Loss

Correct box:
(x', y', w', h')

Object detection은 아니다. 하나의 객체를 찾는것. 기본적으로 image classification과 구조가 비슷.

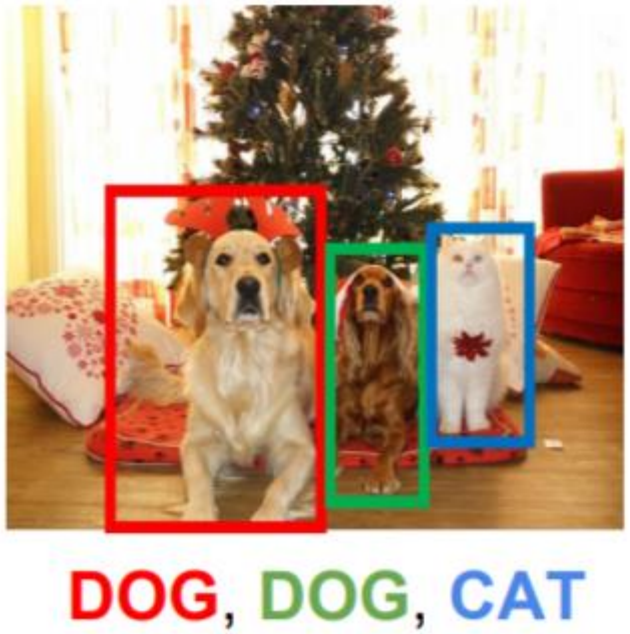
Treat localization as a regression problem!

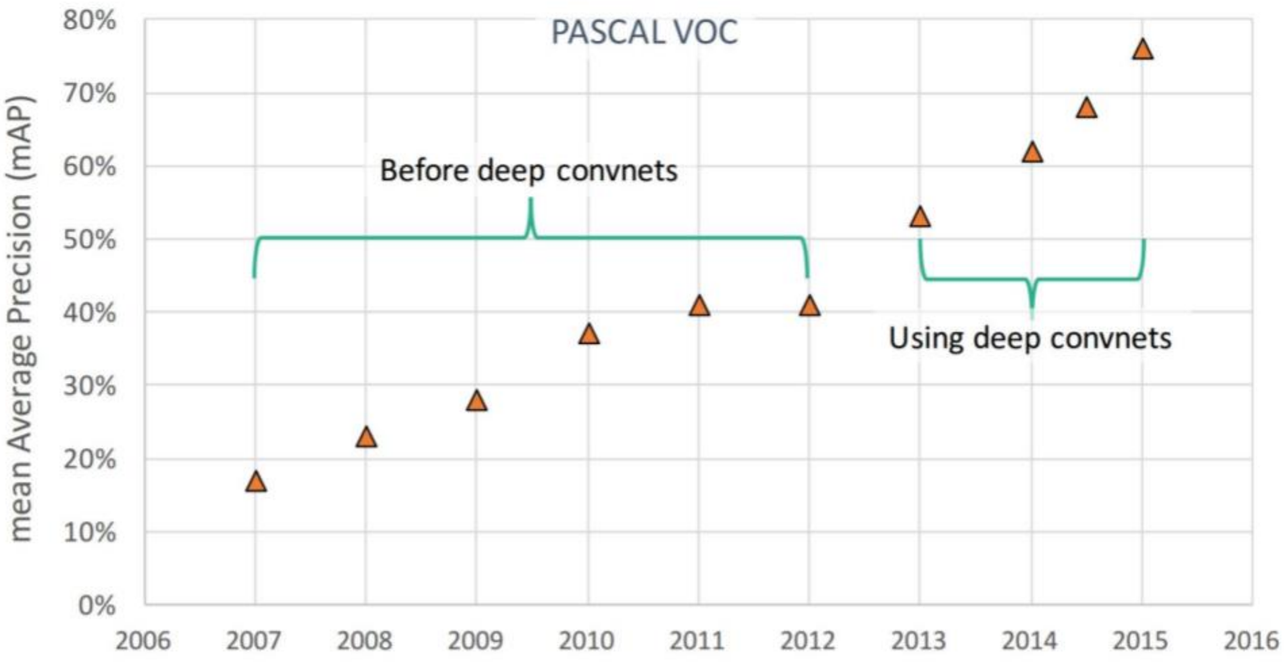
Application: Human Pose Estimation



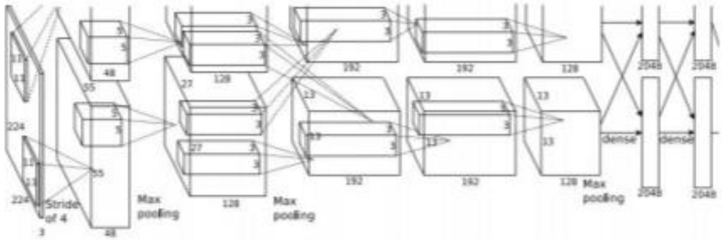
Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

Object Detection

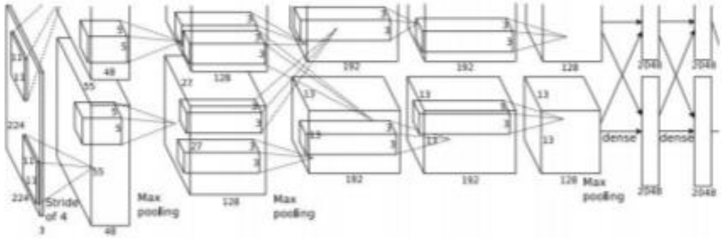




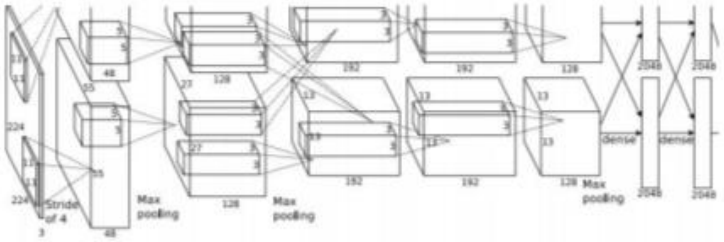
As a regression? – NO!

CAT: (x, y, w, h)   4 numbers

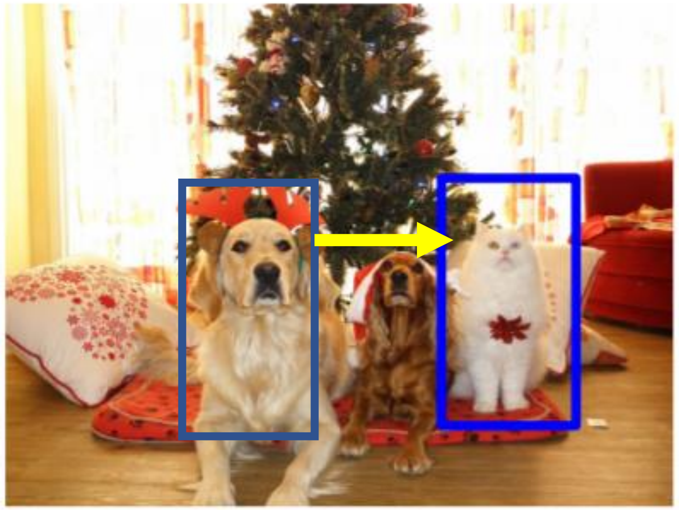
DOG: (x, y, w, h)
DOG: (x, y, w, h)   16 numbers
CAT: (x, y, w, h)

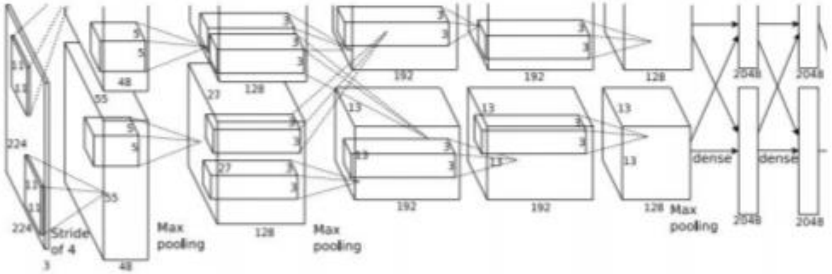



DUCK: (x, y, w, h)   Many
DUCK: (x, y, w, h)   numbers!
....

Sliding window? - NO

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





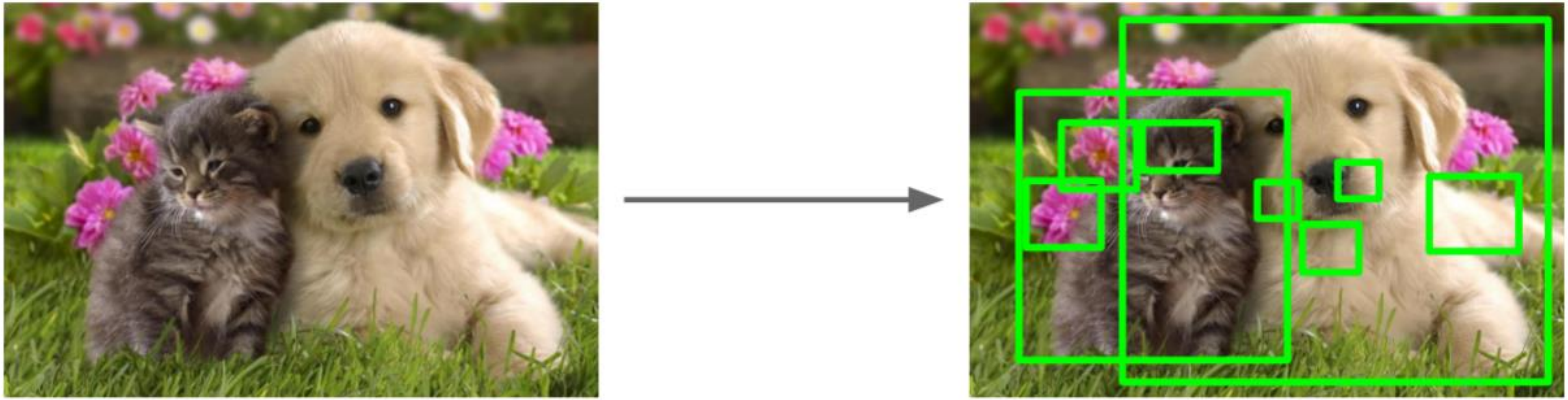
Dog? NO
Cat? YES
Background? NO

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

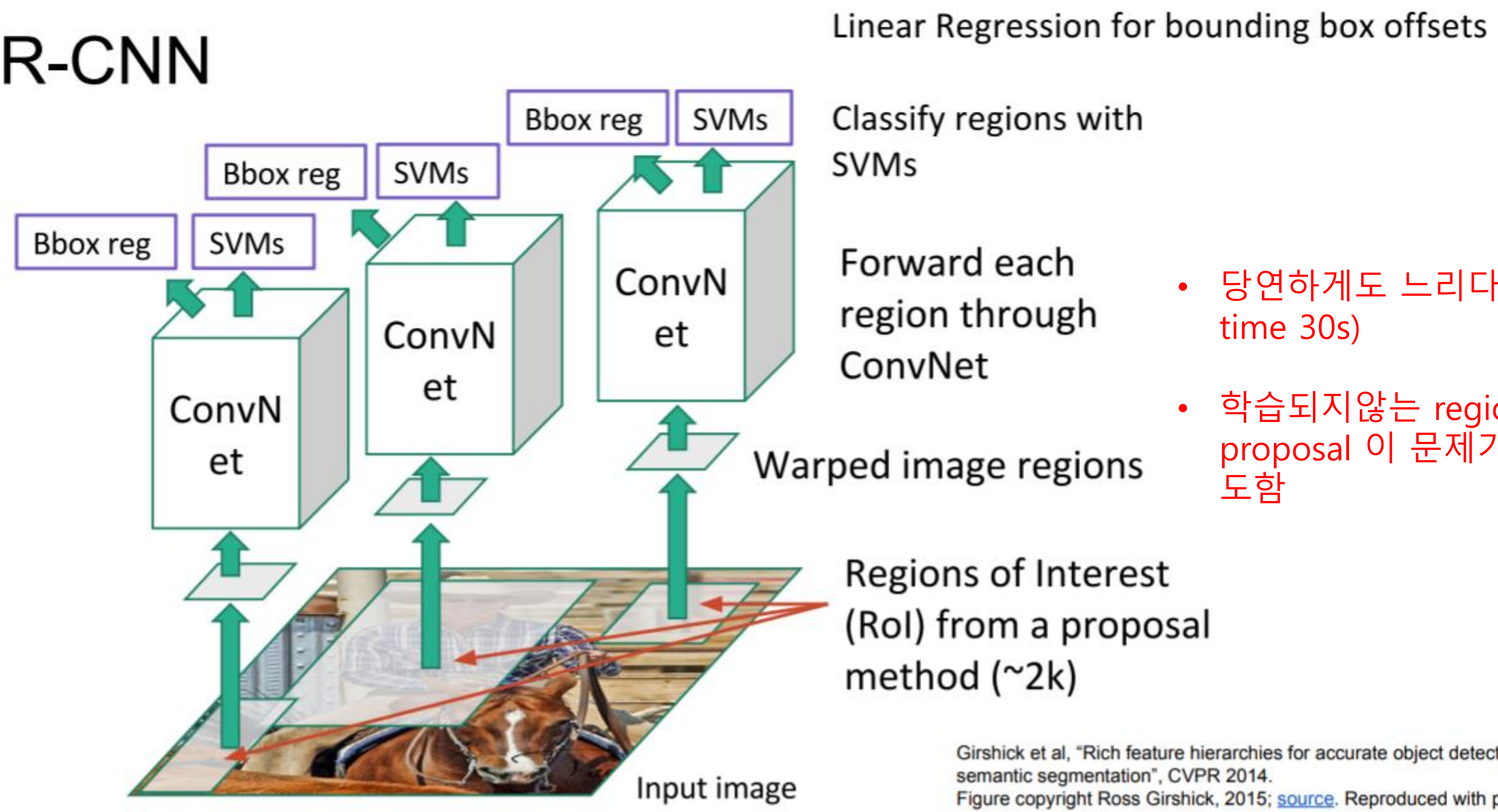
How to decide Crops?

Regional Proposal

- Not deep learning.
- 객체가 있을법한 후보군을 1000개의 selective search로 찾음
- Blobby한곳을 찾아냄.



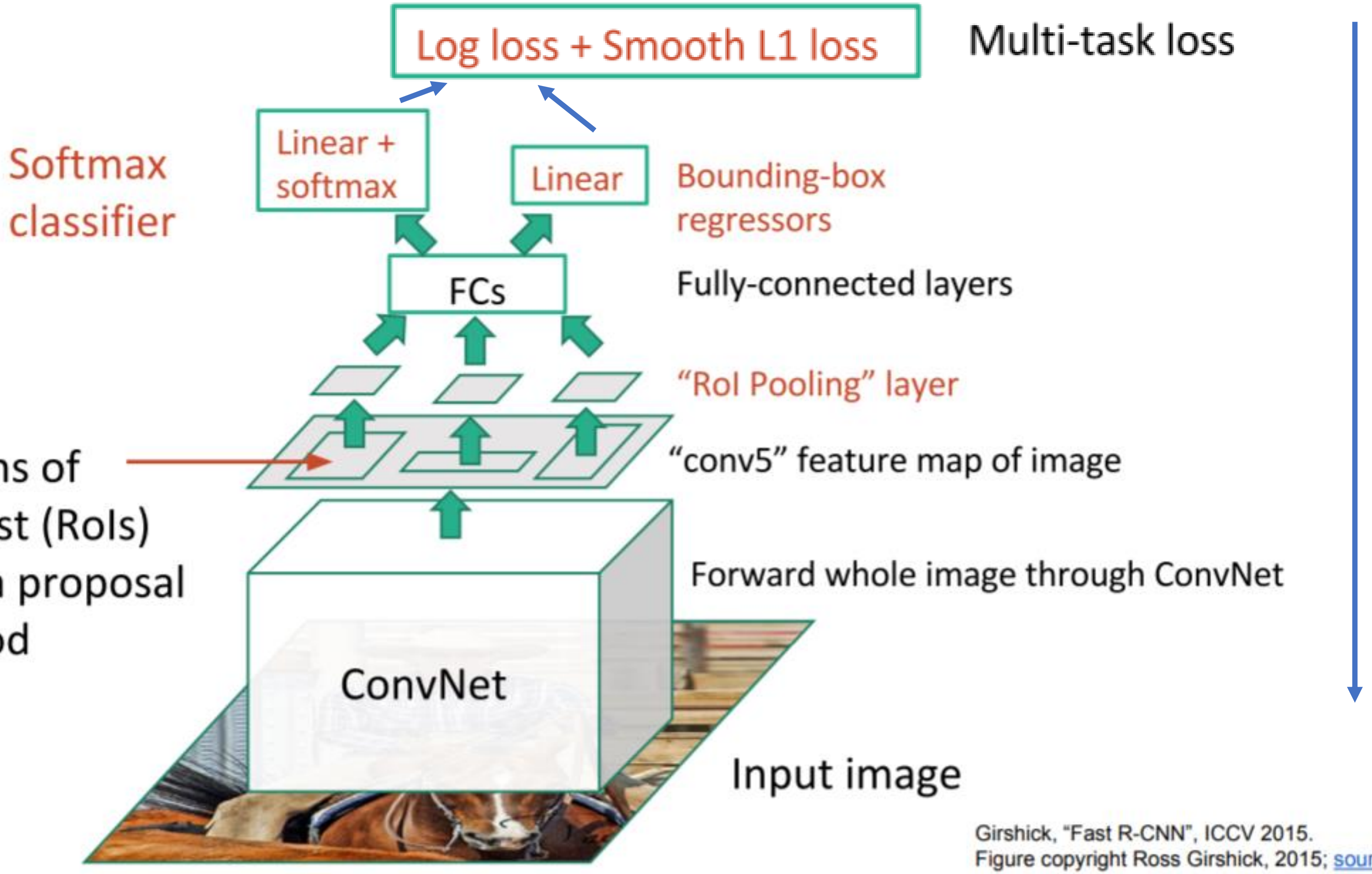
R-CNN



- 당연하게도 느리다. (test time 30s)
- 학습되지않는 regional proposal 이 문제가 되기도함

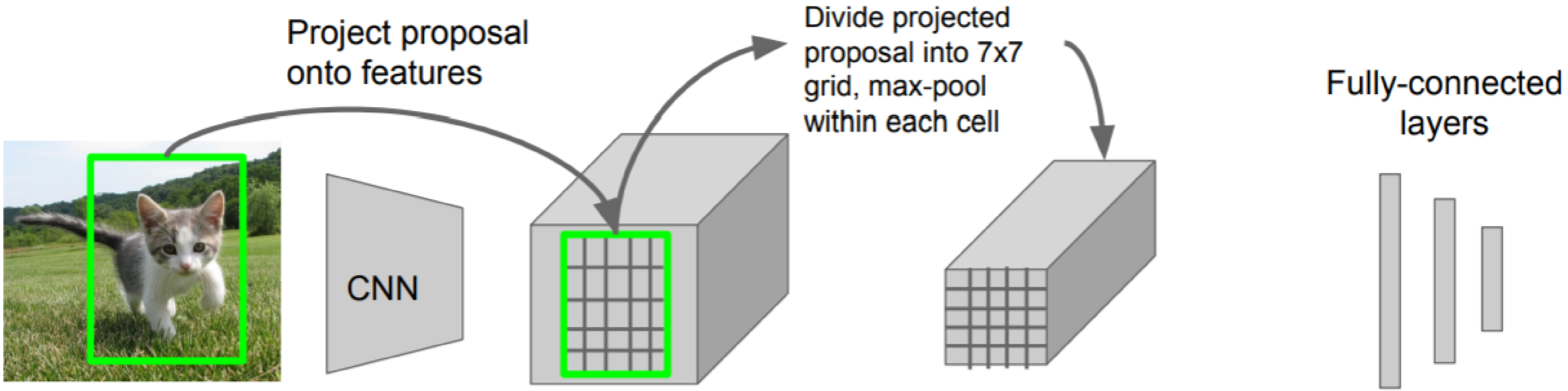
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.

Fast R-CNN



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

Faster R-CNN: RoI Pooling



Hi-res input image:
3 x 640 x 480
with region
proposal

Hi-res conv features:
512 x 20 x 15;

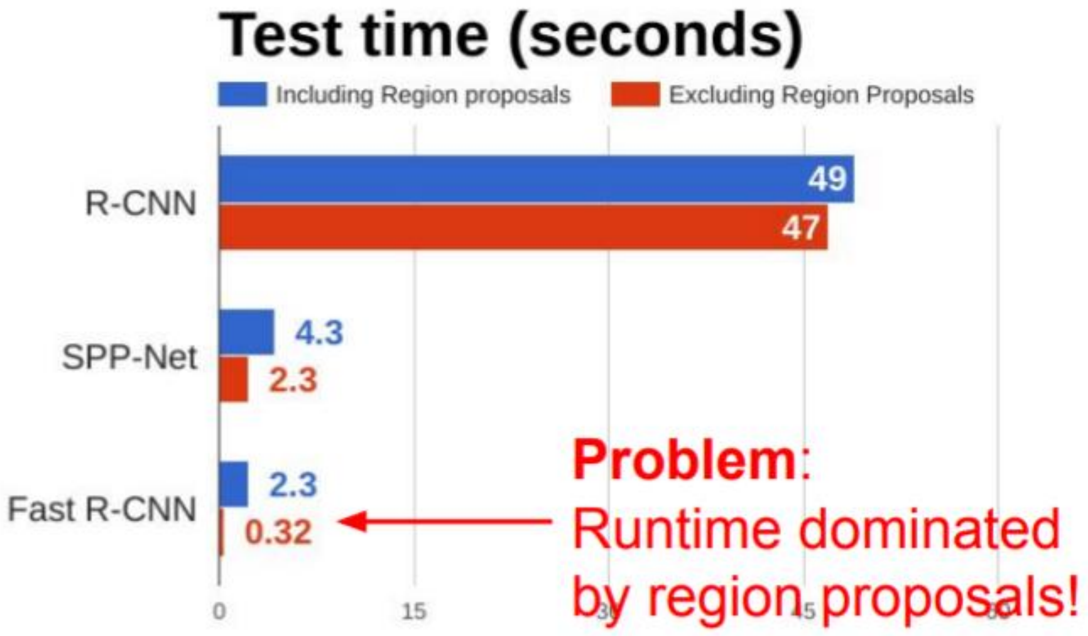
Projected region
proposal is e.g.
512 x 18 x 8
(varies per proposal)

RoI conv features:
512 x 7 x 7
for region proposal

Fully-connected layers expect
low-res conv features:
512 x 7 x 7

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

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

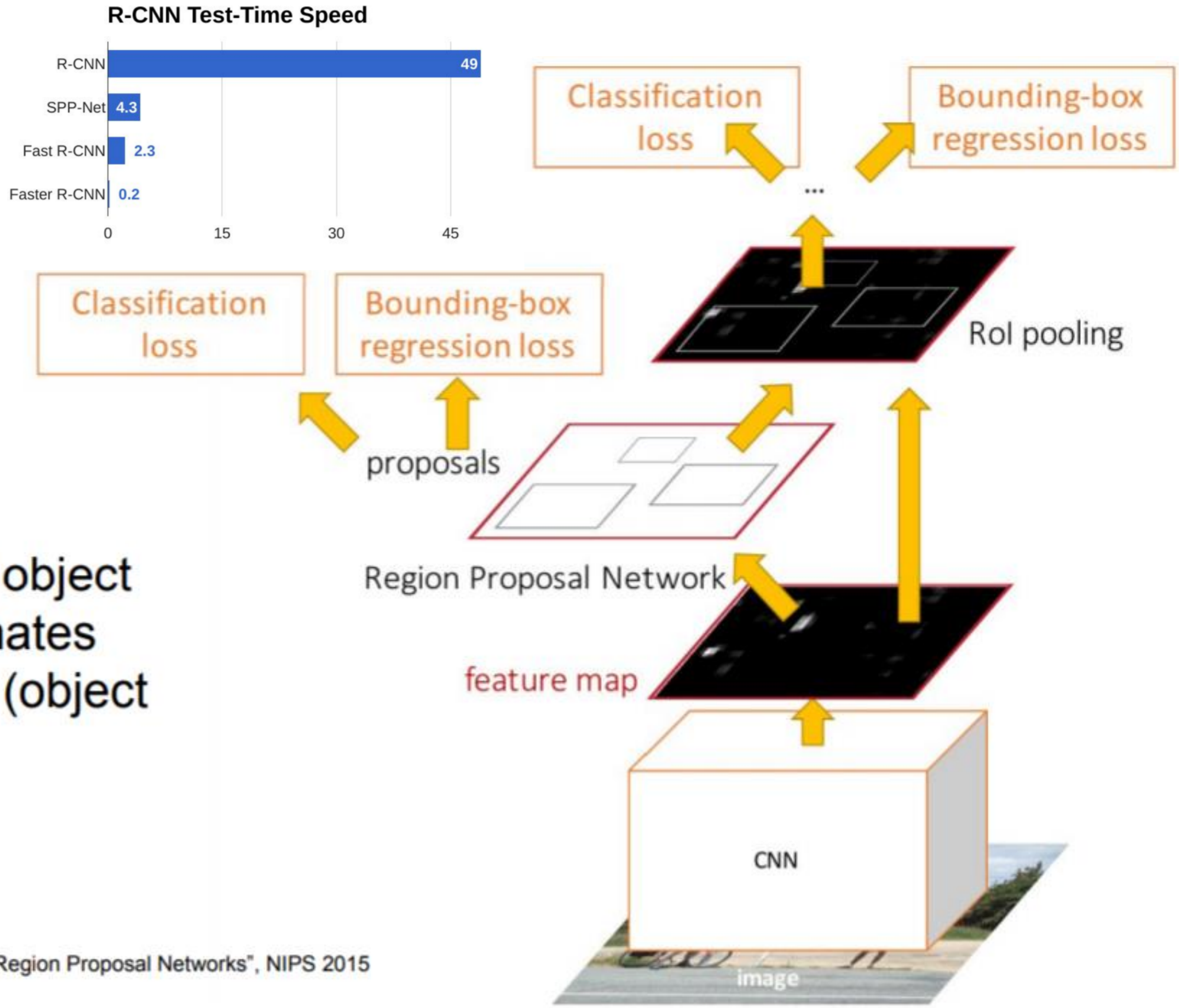
Faster R-CNN:

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

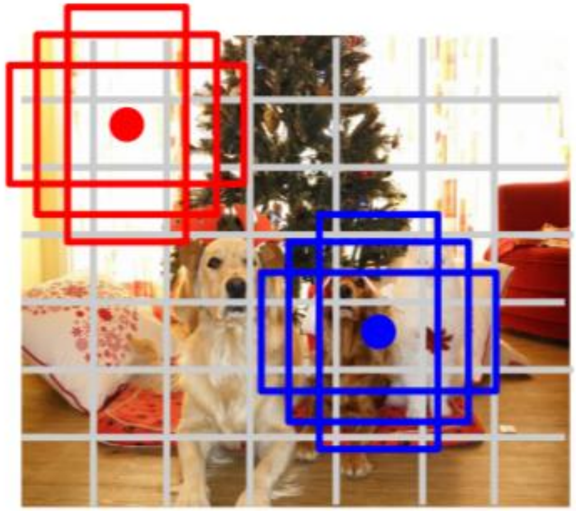


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Detection without proposal: 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

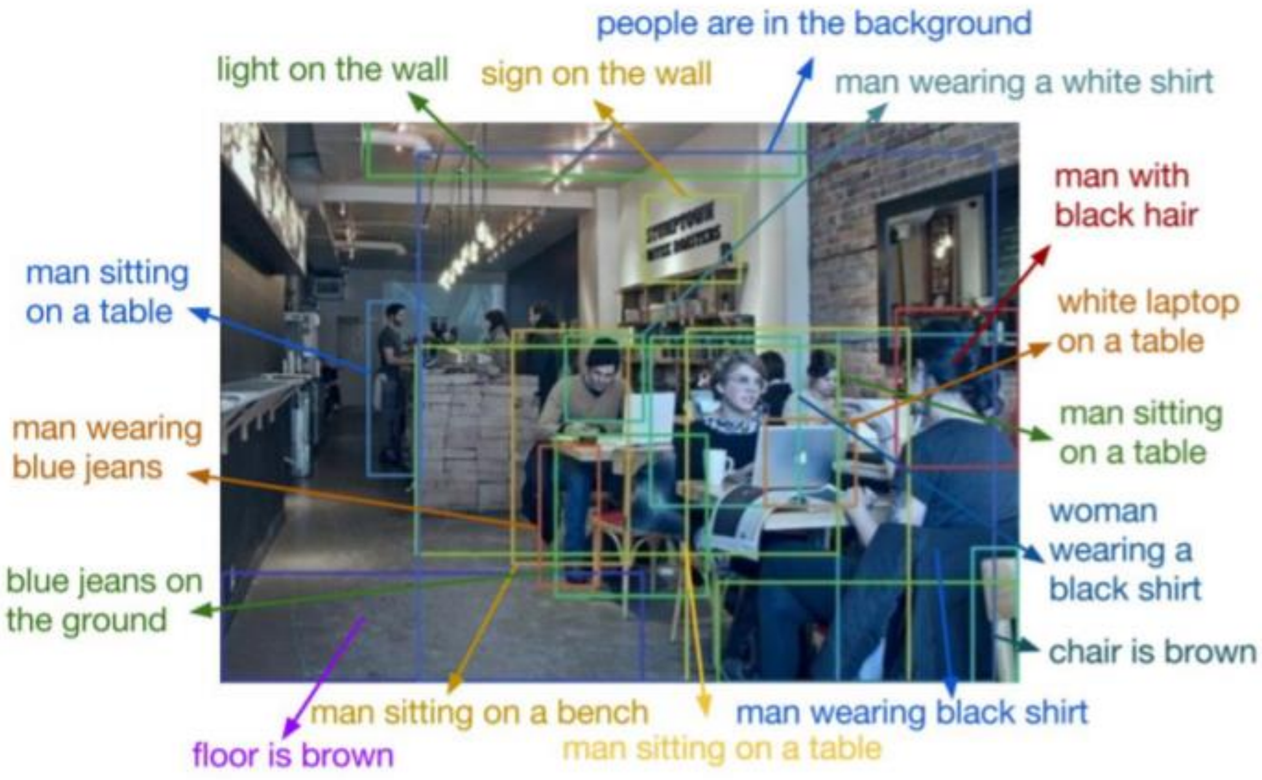
Within each grid cell:

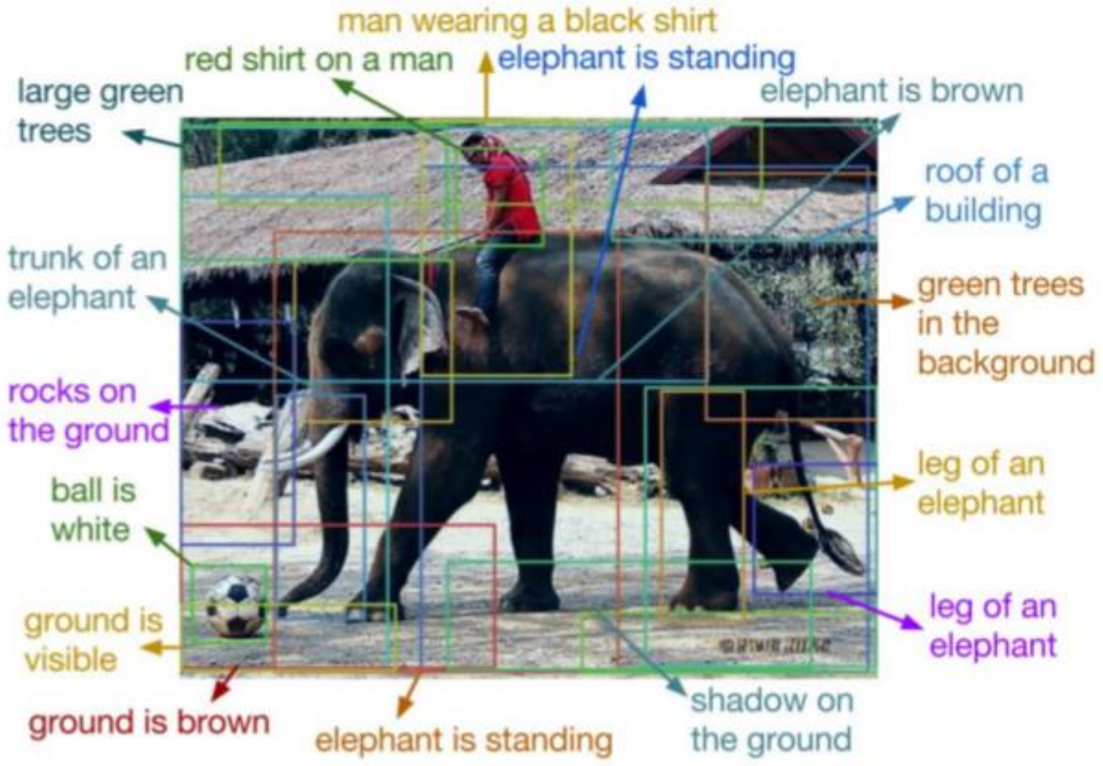
- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

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

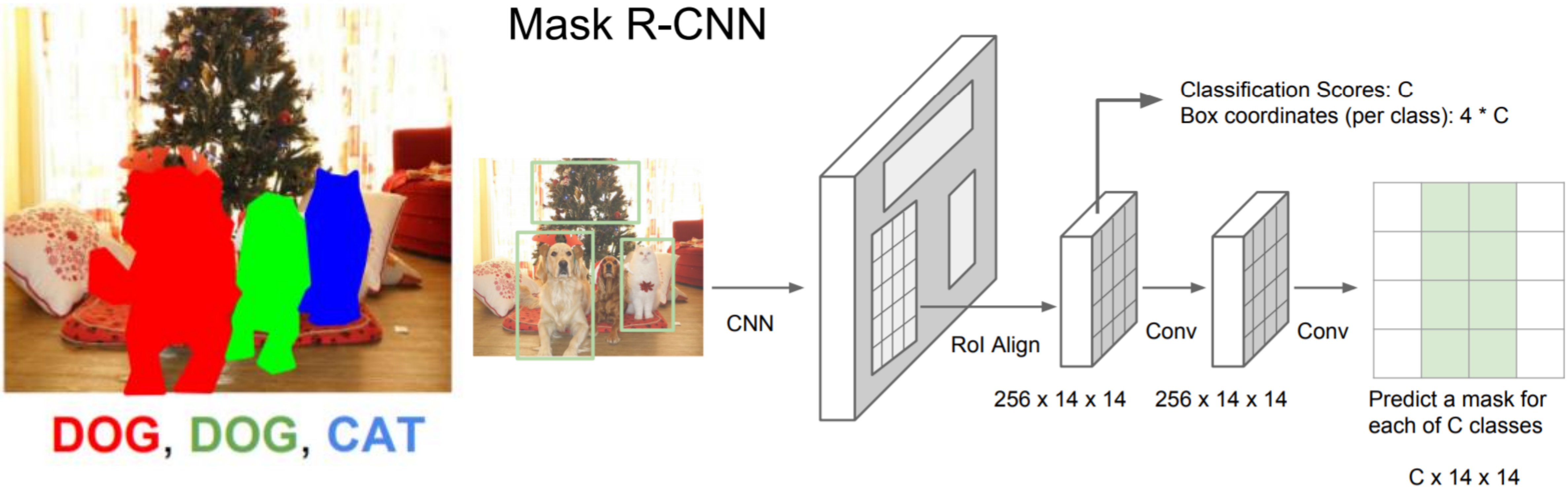
Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016

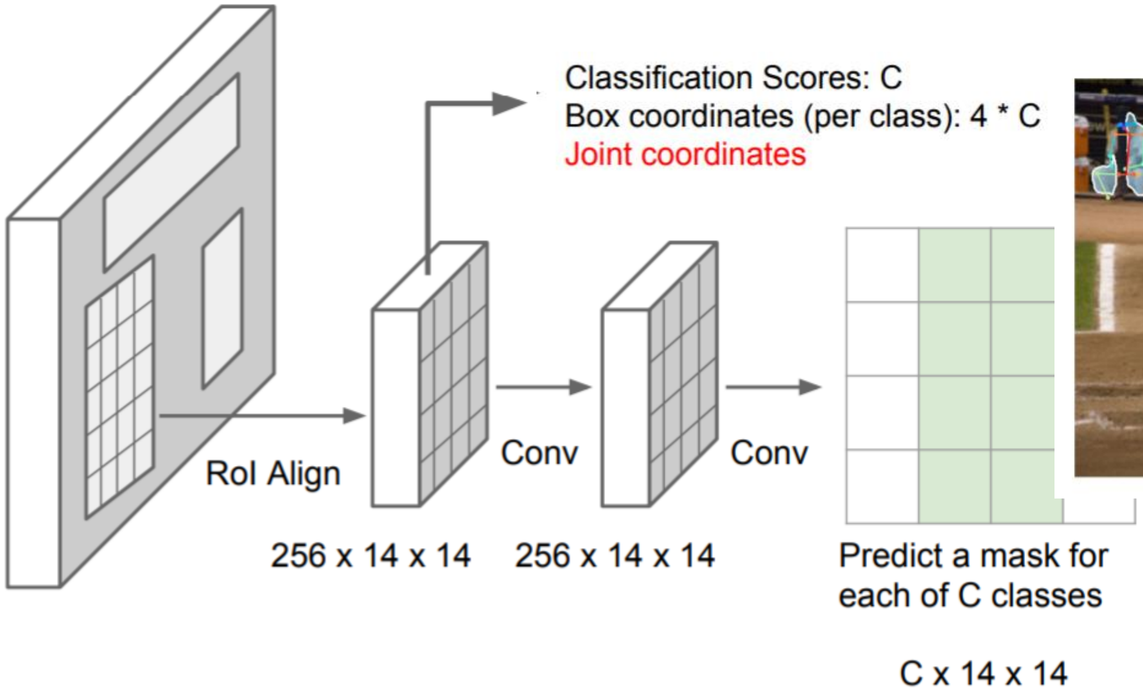
App: Dense Captioning(OD+Captioning)





Instance Segmentation




Classification Scores: C
Box coordinates (per class): 4 * C
Joint coordinates
RoI Align
Conv
Conv
256 x 14 x 14
256 x 14 x 14
Predict a mask for
each of C classes
C x 14 x 14

Thanks

Q&A