



mp

max planck institut
informatik

SIC Saarland Informatics
Campus

High Level Computer Vision

Object Detection and Segmentation
@ May 3, 2023

Bernt Schiele

<https://cms.sic.saarland/hlcvss23/>

**Max Planck Institute for Informatics & Saarland University,
Saarland Informatics Campus Saarbrücken**

So far: Image Classification



This image is CC0 public domain

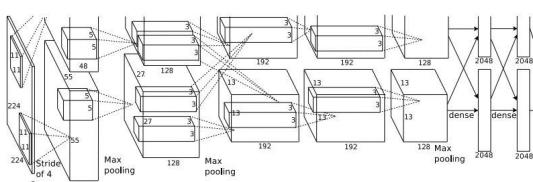


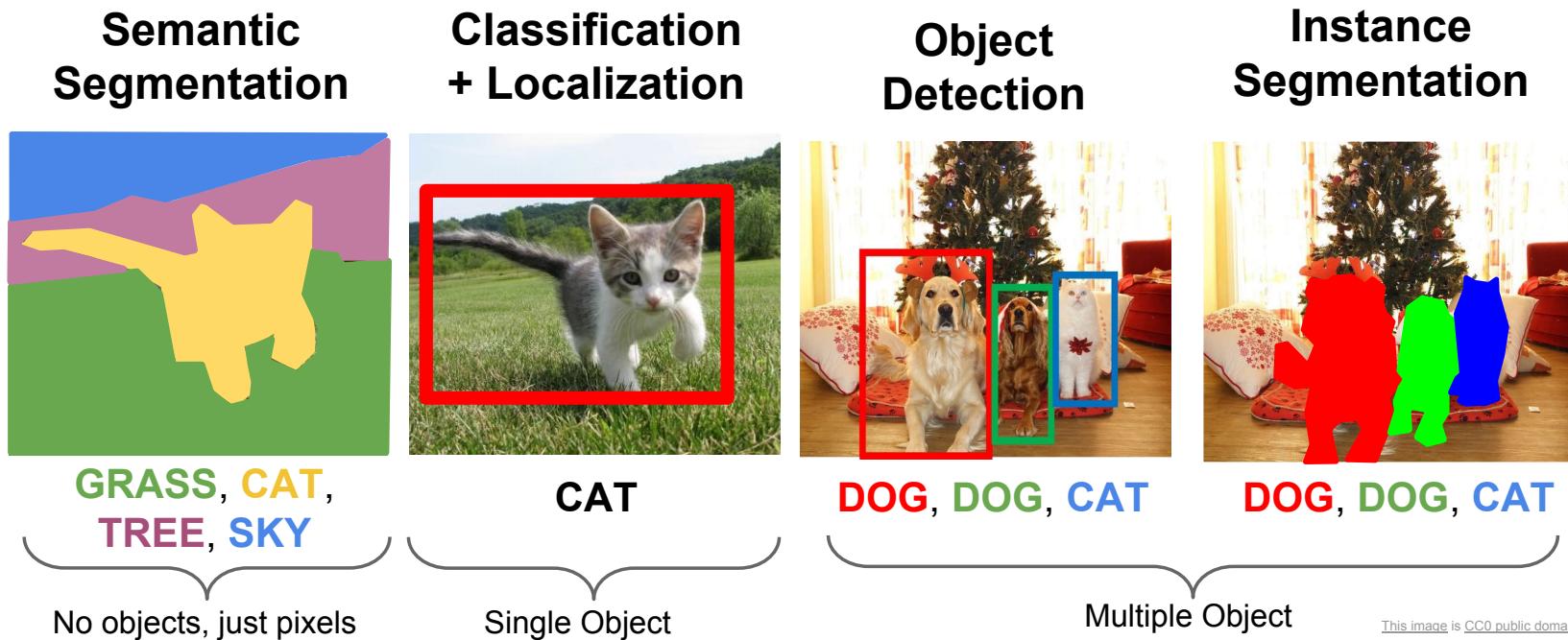
Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Vector:
4096

Fully-Connected:
4096 to 1000

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

Other Computer Vision Tasks



slide credit: Fei-Fei, Justin Johnson, Serena Yeung



Other Computer Vision Tasks — Today

Semantic Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



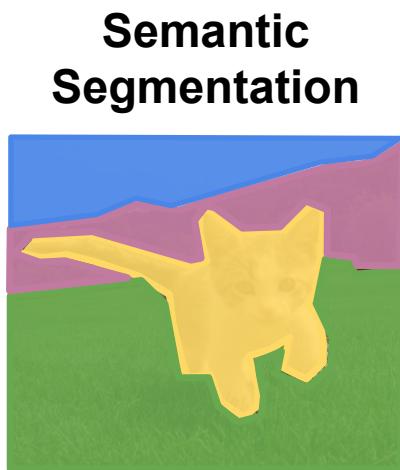
Car

Object categories +
3D bounding boxes

This image is CC0 public domain

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Semantic Segmentation



**GRASS, CAT,
TREE, SKY**

No objects, just pixels

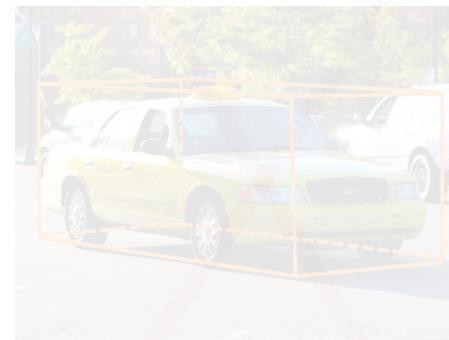
2D Object Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object Detection



Car

Object categories +
3D bounding boxes

This image is CC0 public domain

slide credit: Fei-Fei, Justin Johnson, Serena Yeung



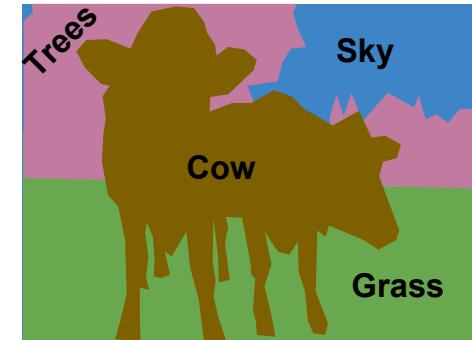
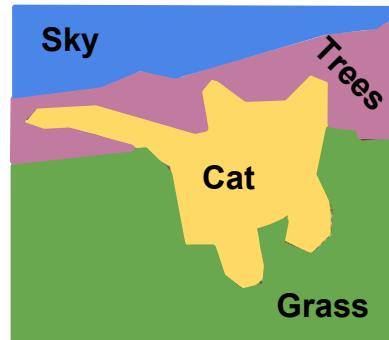
Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels

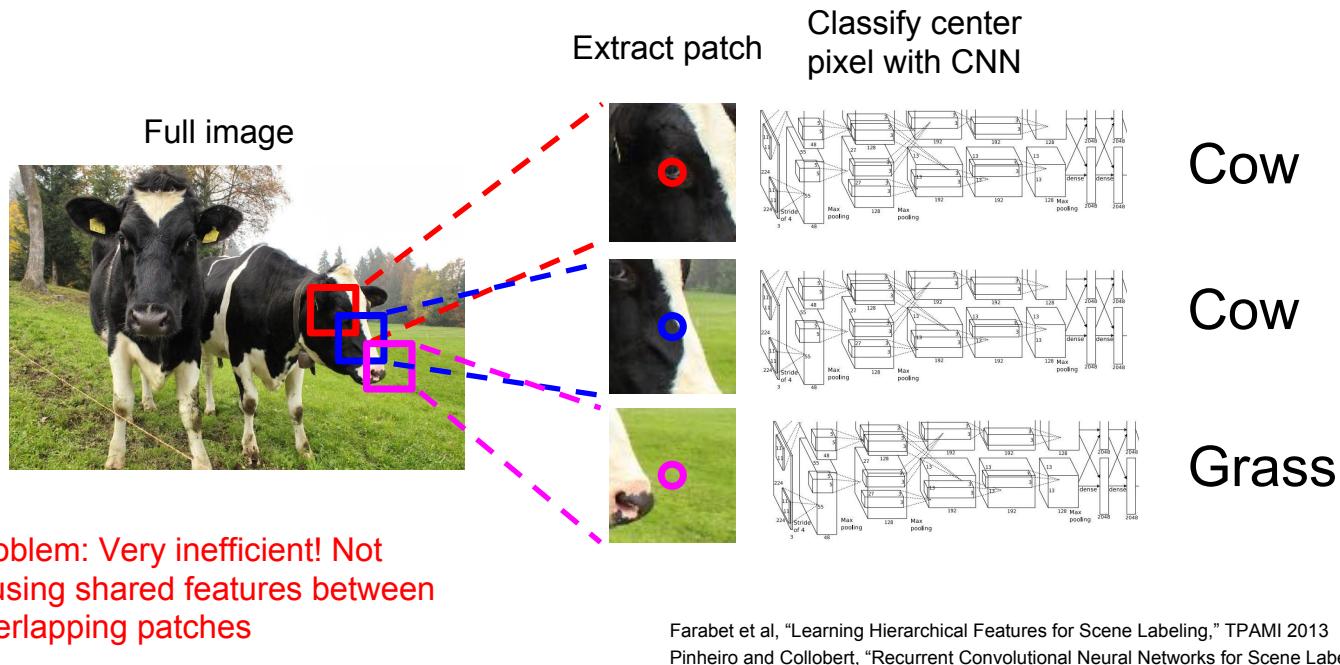


[This image is CC0 public domain](#)



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

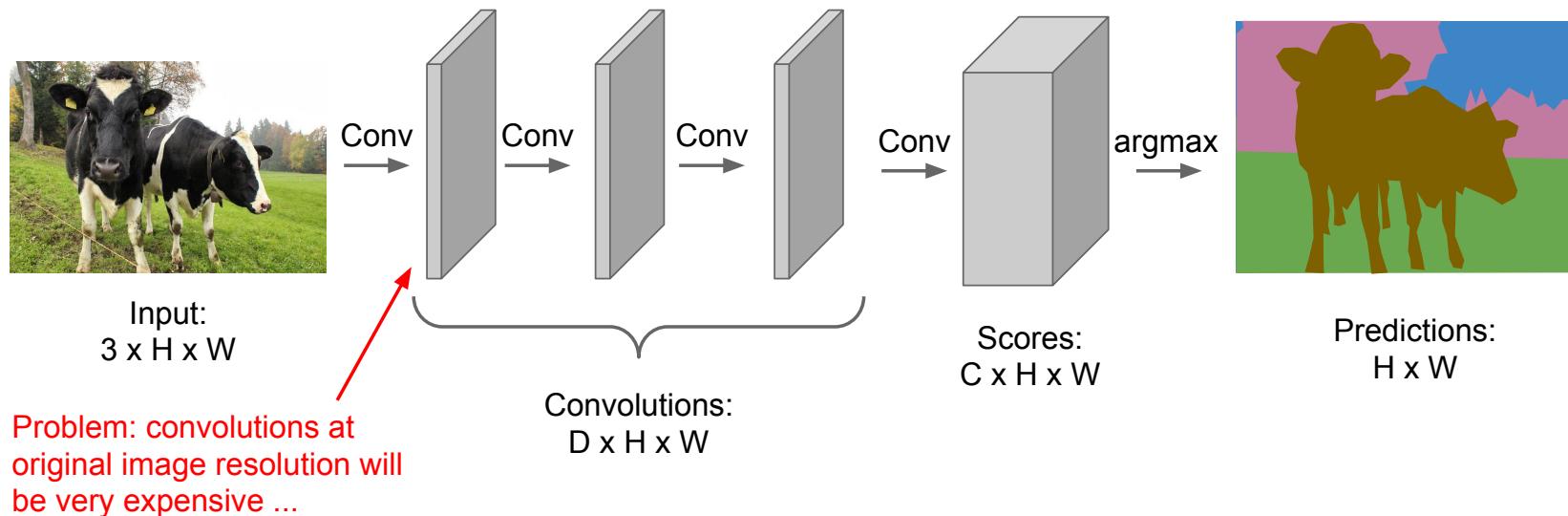
Semantic Segmentation Idea: Sliding Window



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

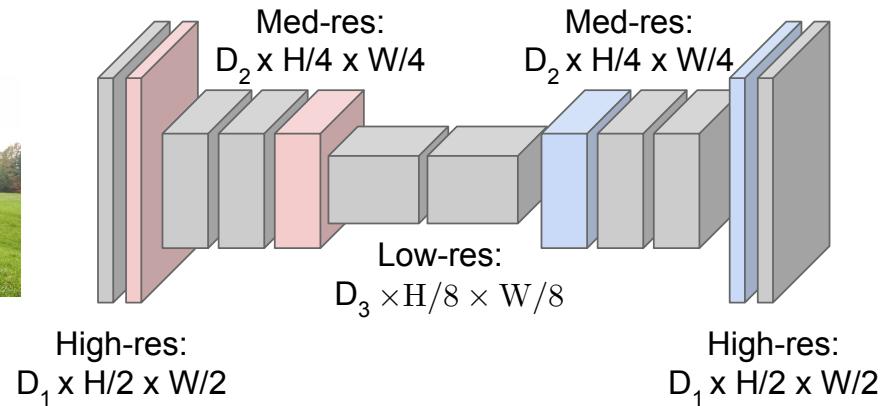


Semantic Segmentation Idea: Fully Convolutional

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



Input:
 $3 \times H \times W$



Predictions:
 $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

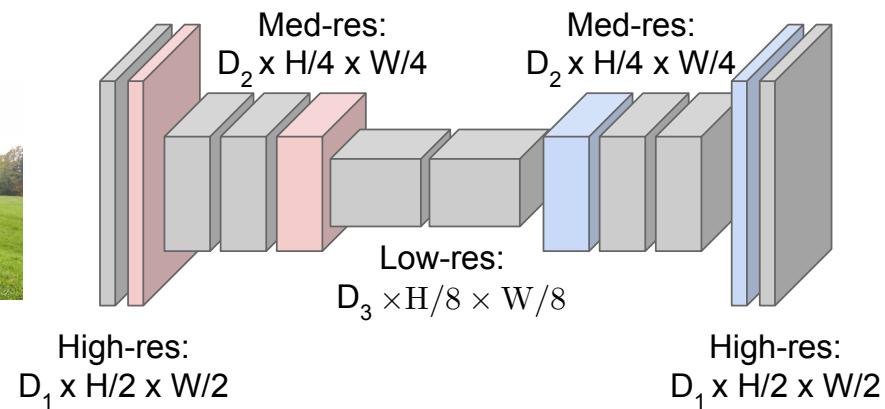
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided
convolution

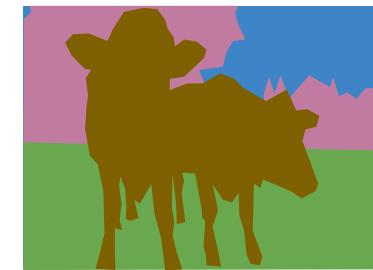


Input:
 $3 \times H \times W$

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



Upsampling:
???



Predictions:
 $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

In-Network Upsampling: "Unpooling"

Nearest Neighbor

1	2
3	4



1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Input: 2 x 2

Output: 4 x 4

"Bed of Nails"

1	2
3	4



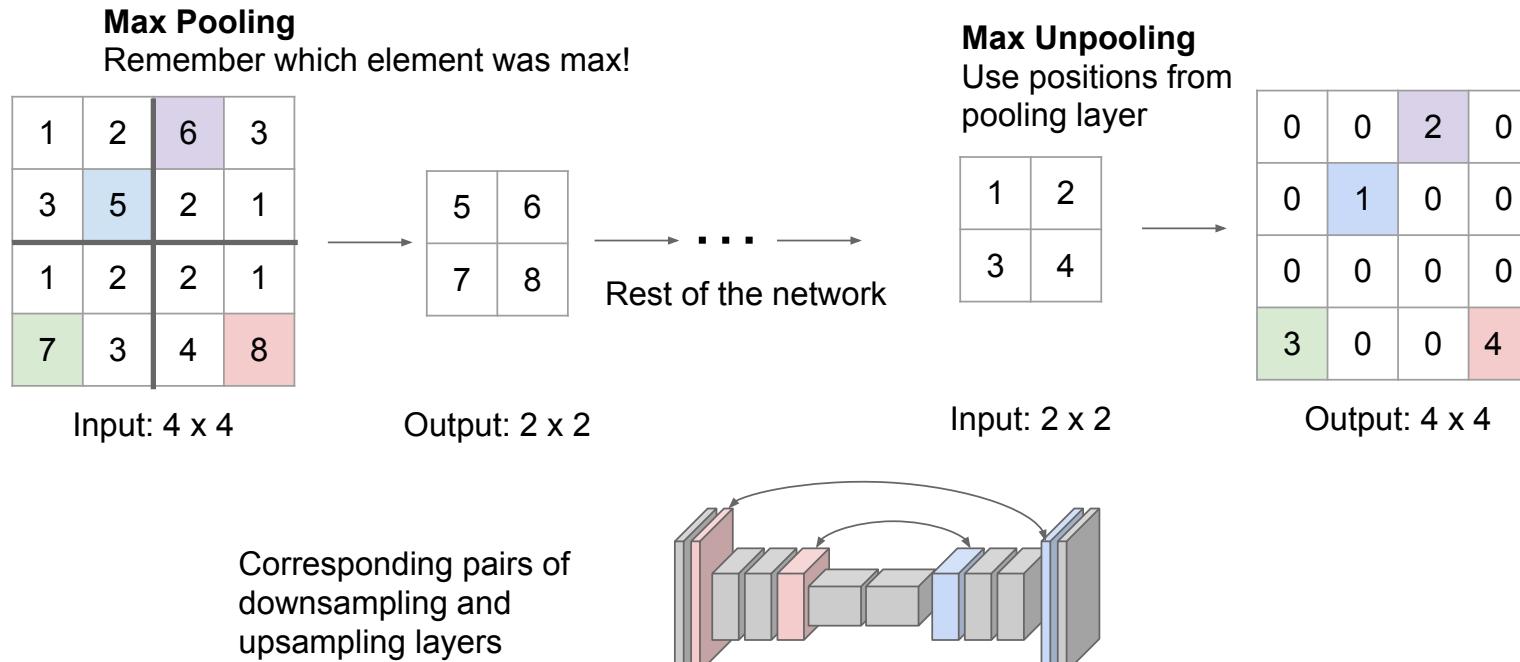
1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Input: 2 x 2

Output: 4 x 4

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

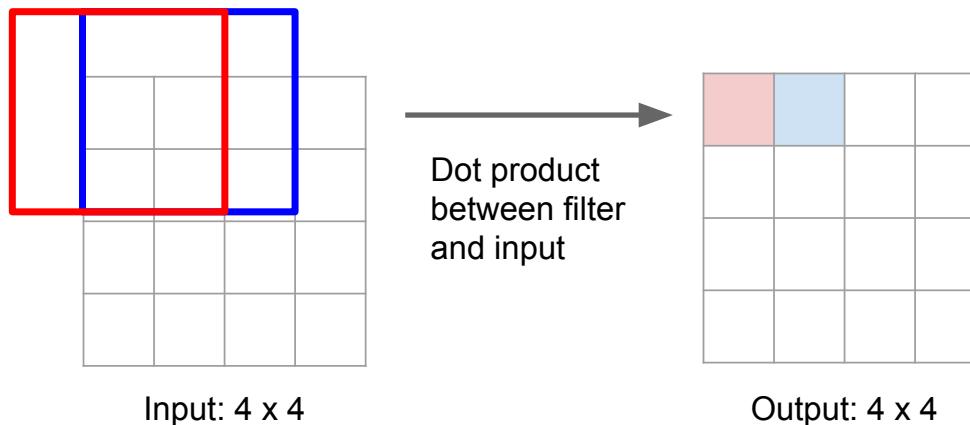
In-Network Upsampling: "Max Unpooling"



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Learnable Upsampling: Transpose Convolution

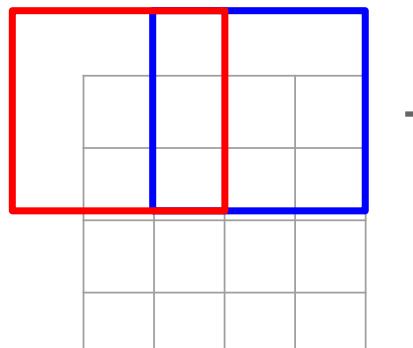
Recall: Normal 3×3 convolution, stride 1 pad 1



slide credit: Fei-Fei, Justin Johnson, Serena Yeung

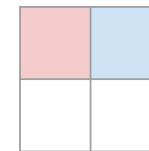
Learnable Upsampling: Transpose Convolution

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



Input: 4×4

Dot product
between filter
and input



Output: 2×2

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

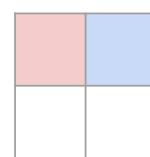
Stride gives ratio between
movement in input and
output

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Learnable Upsampling: Transpose Convolution

Other names:

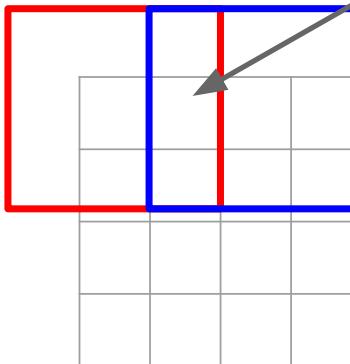
- Deconvolution (bad)
- Upconvolution
- Fractionally strided convolution
- Backward strided convolution



Input: 2 x 2

3 x 3 transpose convolution, stride 2 pad 1

Input gives weight for filter



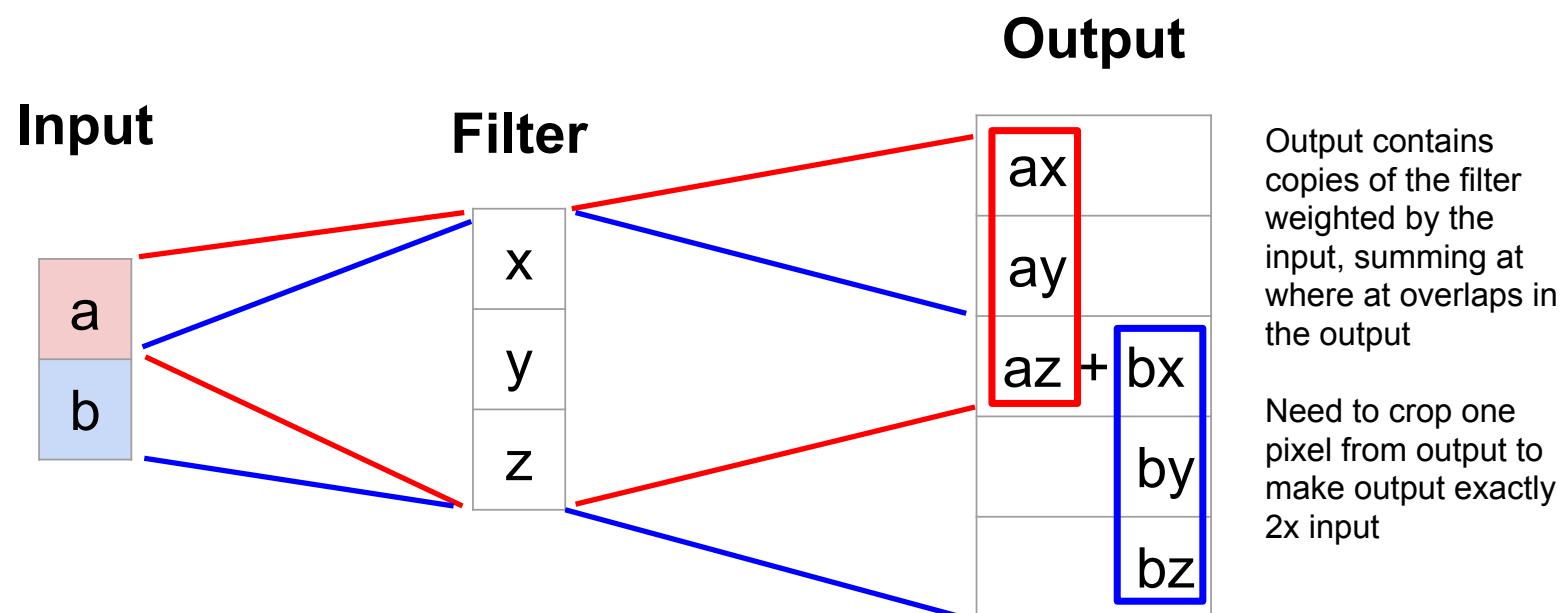
Output: 4 x 4

Sum where output overlaps

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

Stride gives ratio between movement in output and input

Learnable Upsampling: 1D Example



slide credit: Fei-Fei, Justin Johnson, Serena Yeung



Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

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

Example: 1D conv, kernel size=3, stride=1, padding=1 **1D input
(e.g. image)**

**1D Convolution Kernel
expanded into Matrix**

Convolution transpose multiplies by the transpose of the same matrix:

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

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung



Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

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

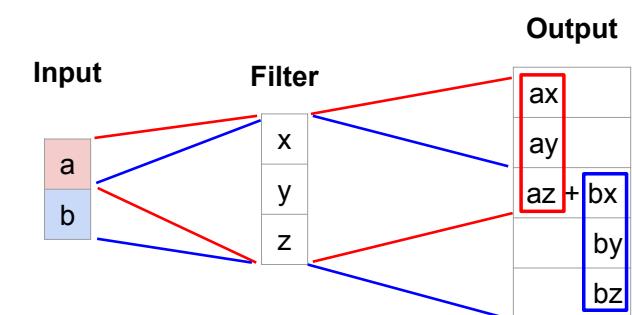
$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & 0 & x & y & z & 0 \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ bx + cy + dz \end{bmatrix}$$

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

Convolution transpose multiplies by the transpose of the same matrix:

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

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



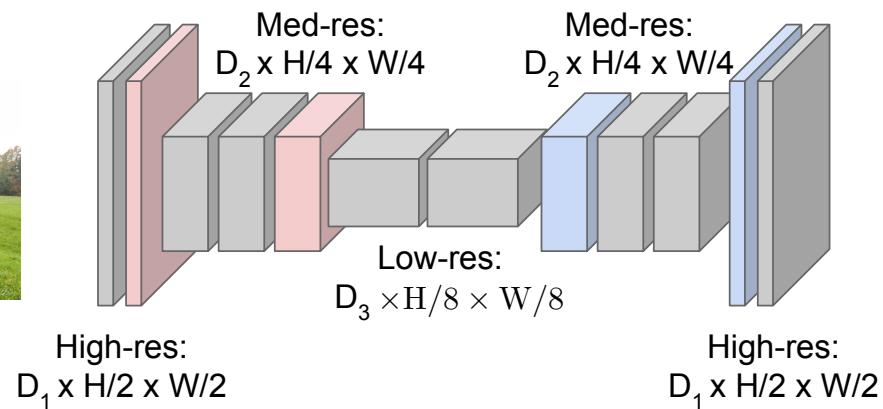
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution



Input:
 $3 \times H \times W$

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



Upsampling:
Unpooling or strided transpose convolution



Predictions:
 $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

2D Object Detection

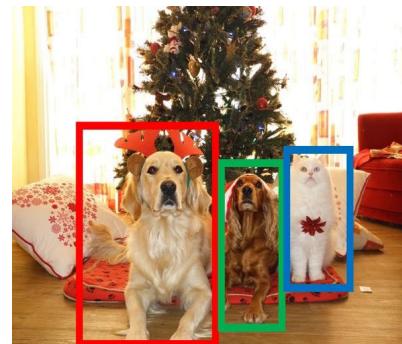
Semantic
Segmentation



GRASS, CAT,
TREE, SKY

No objects, just pixels

2D Object
Detection



DOG, DOG, CAT

Object categories +
2D bounding boxes

3D Object
Detection



Car

Object categories +
3D bounding boxes

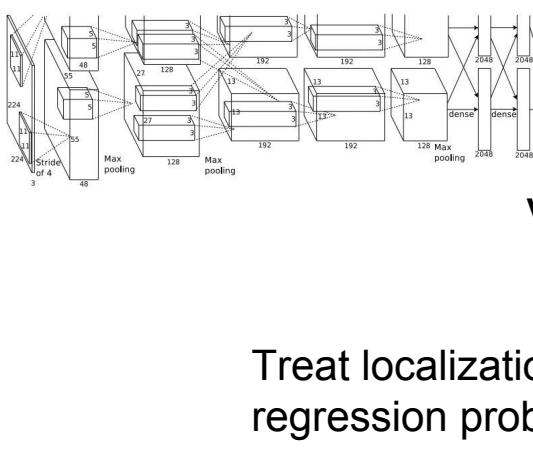
This image is CC0 public domain

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Classification + Localization



This image is CC0 public domain



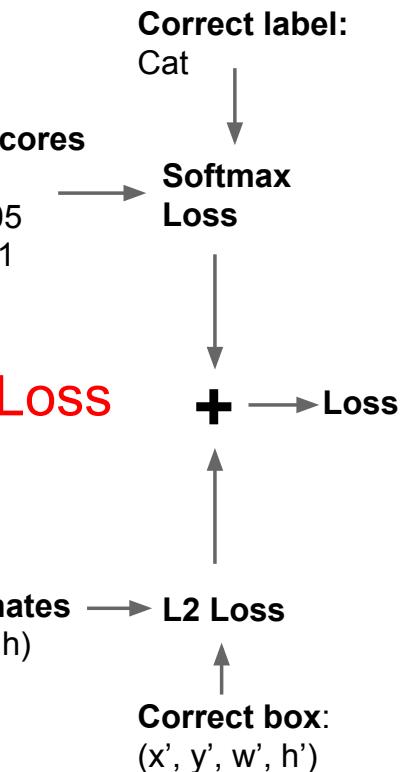
Treat localization as a
regression problem!

Vector: Fully
Connected:
4096 to 4096

Fully
Connected:
4096 to 1000

Multitask Loss

Box
Coordinates → L2 Loss
(x, y, w, h)

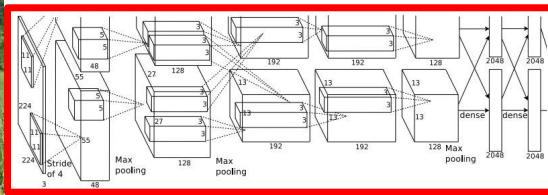


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Classification + Localization



This image is CC0 public domain



Often pretrained on ImageNet
(Transfer learning)

Treat localization as a
regression problem!

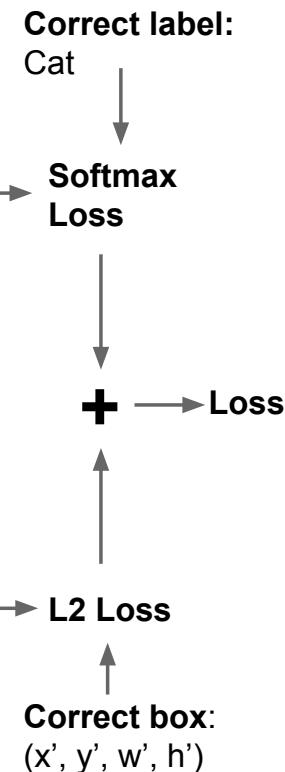
Fully Connected:
4096 to 1000

Vector: 4096

Fully Connected:
4096 to 4

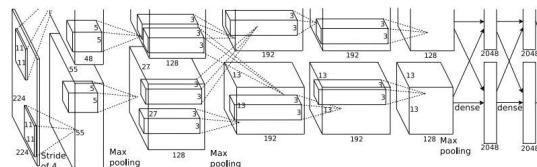
Class Scores
Cat: 0.9
Dog: 0.05
Car: 0.01
...

Box Coordinates → L2 Loss
(x, y, w, h)

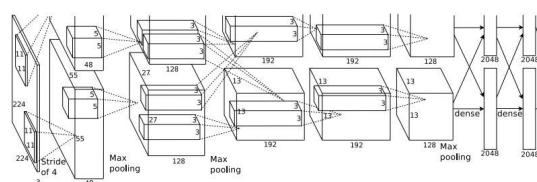


slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Object Detection as Regression?



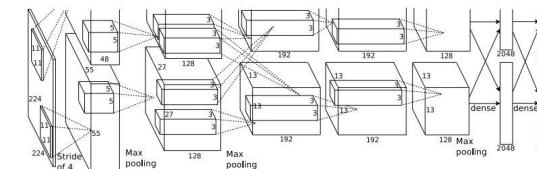
CAT: (x, y, w, h)



DOG: (x, y, w, h)

DOG: (x, y, w, h)

CAT: (x, y, w, h)



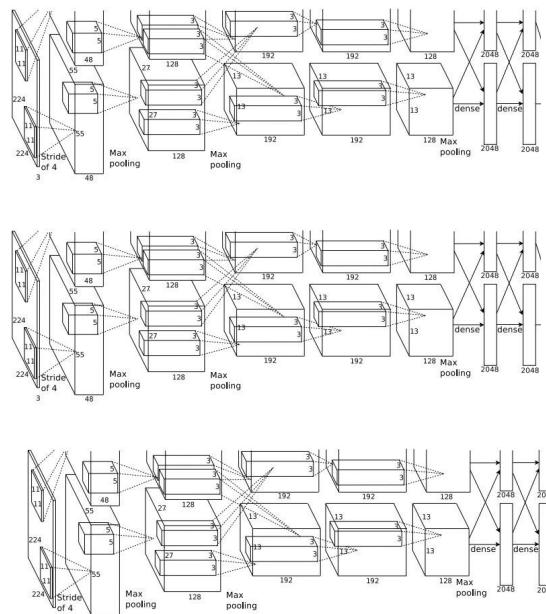
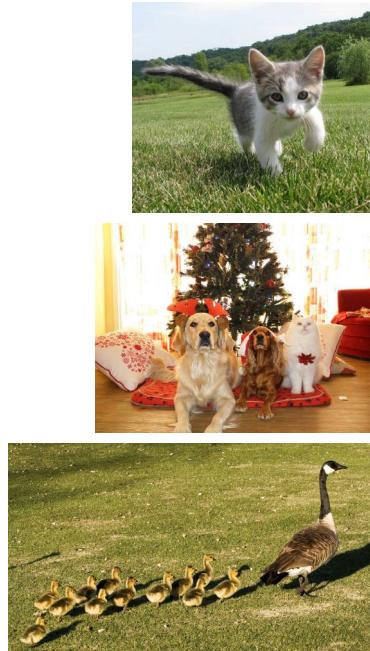
DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

...

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Object Detection as Regression?



Each image needs a
different number of outputs!

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

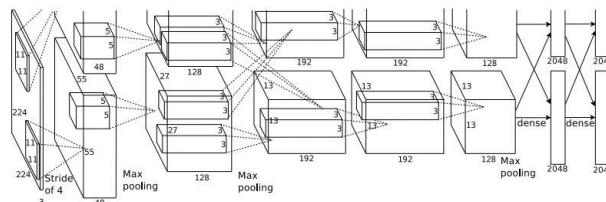
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Object Detection as Classification: Sliding Window

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

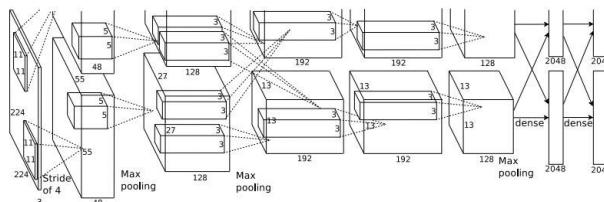


Dog? NO
Cat? NO
Background? YES

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Object Detection as Classification: Sliding Window

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

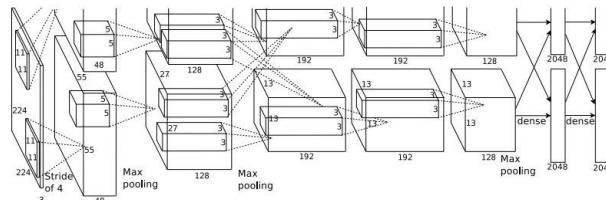


Dog? YES
Cat? NO
Background? NO

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Object Detection as Classification: Sliding Window

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

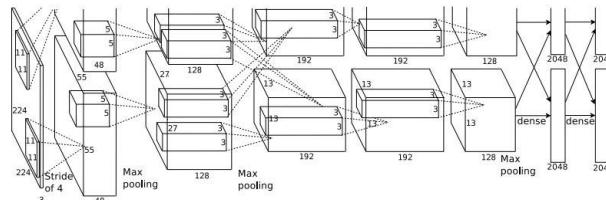


Dog? YES
Cat? NO
Background? NO

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Object Detection as Classification: Sliding Window

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

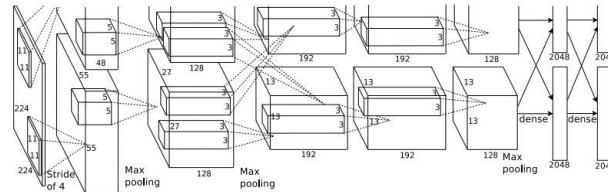
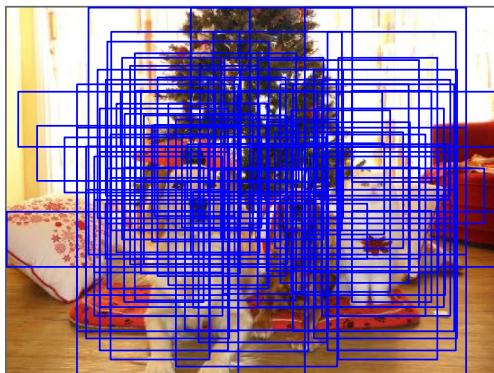


Dog? NO
Cat? YES
Background? NO

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Object Detection as Classification: Sliding Window

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



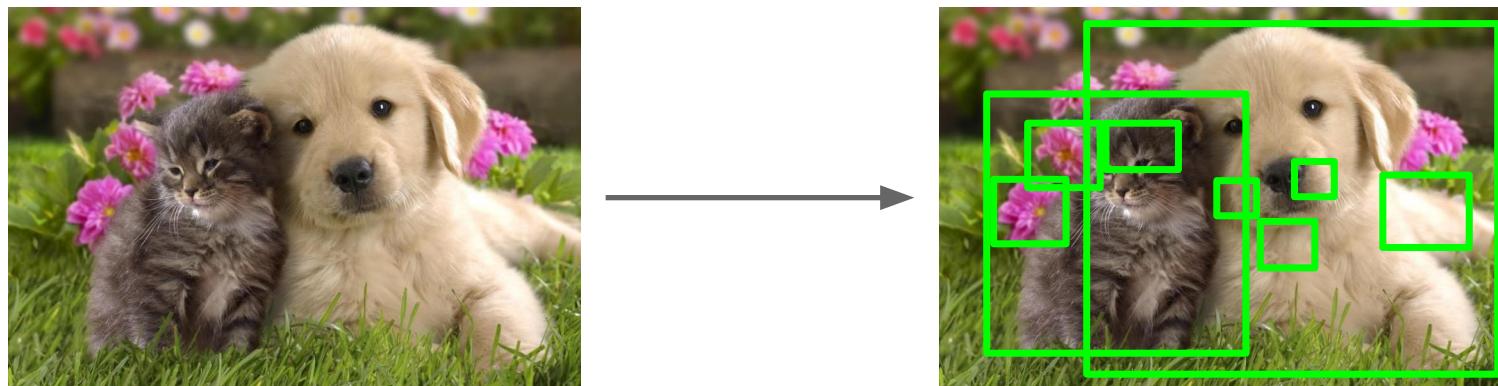
Dog? NO
Cat? YES
Background? NO

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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Region Proposals / e.g. Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 2000 region proposals in a few seconds on CPU



Alexe et al, "Measuring the objectness of image windows", TPAMI 2012

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Region Proposal Step

Segmentation as Selective Search for Object Recognition

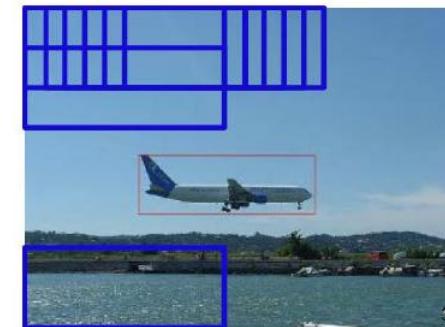
K. van de Sande¹, J. Uijlings², T. Gevers¹, and A. Smeulders¹

University of Amsterdam¹ and University of Trento²

Selective Search: Motivation

Viola IJCV 2004
Dalal CVPR 2005
Felzenszwalb TPAMI 2010
Vedaldi ICCV 2009

- Many approaches (at the time) use exhaustive search:
 - ▶ visit **every** location in an image
 - ▶ problem: computationally expensive:
 - number of possible locations cannot be too large
-> number of grid locations & aspect ratio(s) need to be small
 - evaluation cost per location should be low
-> simple features / classifiers
- to go beyond this - we should aim for something more “sophisticated”



Selective Search: Main Design Criteria

- **High recall**
 - We do not want to lose any objects, since they cannot be recovered later.
- Coarse locations are sufficient
 - Accurate delineation is not necessary for recognition
 - In contrary, nearby context might be useful
-> use bounding boxes
- Fast to compute
 - Necessary when operating with large datasets
-> <10s/image

Selective Search: How to Obtain High Recall?

- Images are intrinsically hierarchical



- Segmentation at single scale are not enough
-> hypotheses based on hierarchical grouping

Selective Search: Method

- Start by oversegmenting the input image



“Efficient graph-based image segmentation”
Felzenszwalb and Huttenlocher, IJCV 2004

Selective Search: Method

- compute similarity measure between all adjacent region pairs a and b (e.g.) as:

$$S(a, b) = \alpha S_{size}(a, b) + \beta S_{color}(a, b)$$

- ▶ with

$$S_{size}(a, b) = 1 - \frac{\text{size}(a) + \text{size}(b)}{\text{size}(image)}$$

encourages small regions to merge early

- ▶ and

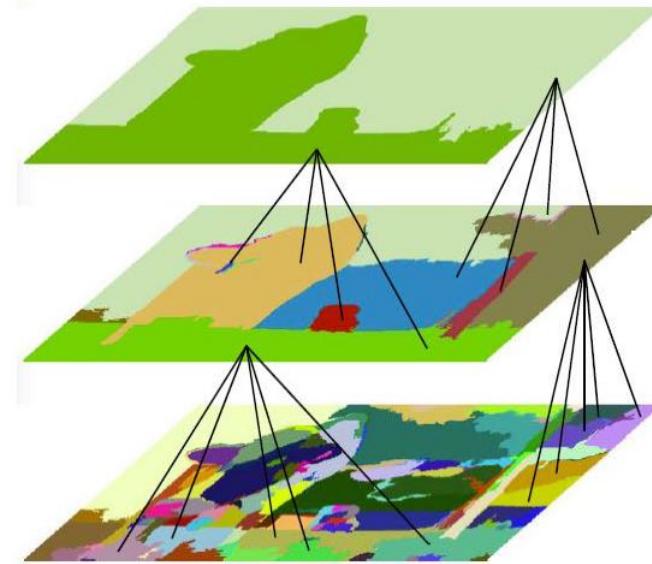
$$S_{color}(a, b) = \sum_{k=1}^n \min(a^k, b^k)$$

a^k, b^k are color histograms, encouraging “similar (color)” regions to merge

- ▶ for slightly more elaborated similarities see their IJCV-paper

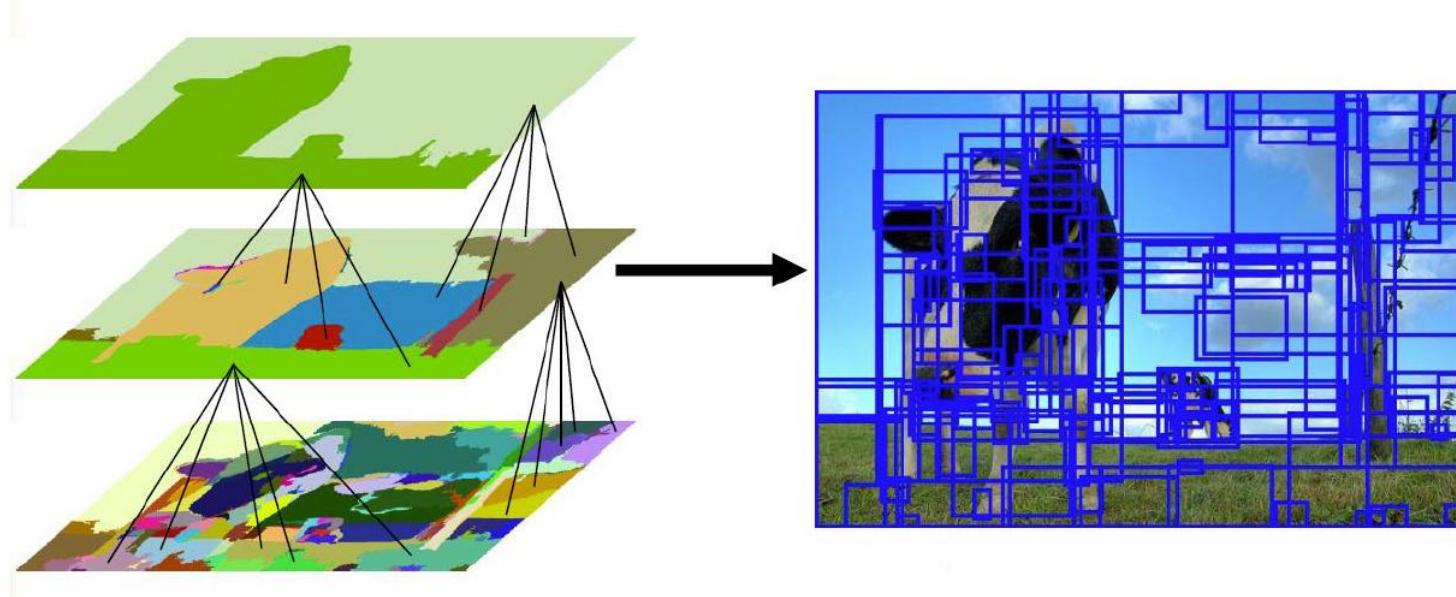
Selective Search: Method

1. Merge two most similar regions based on S .
2. Update similarities between the new region and its neighbors.
3. Go back to step 1. until the whole image is a single region.

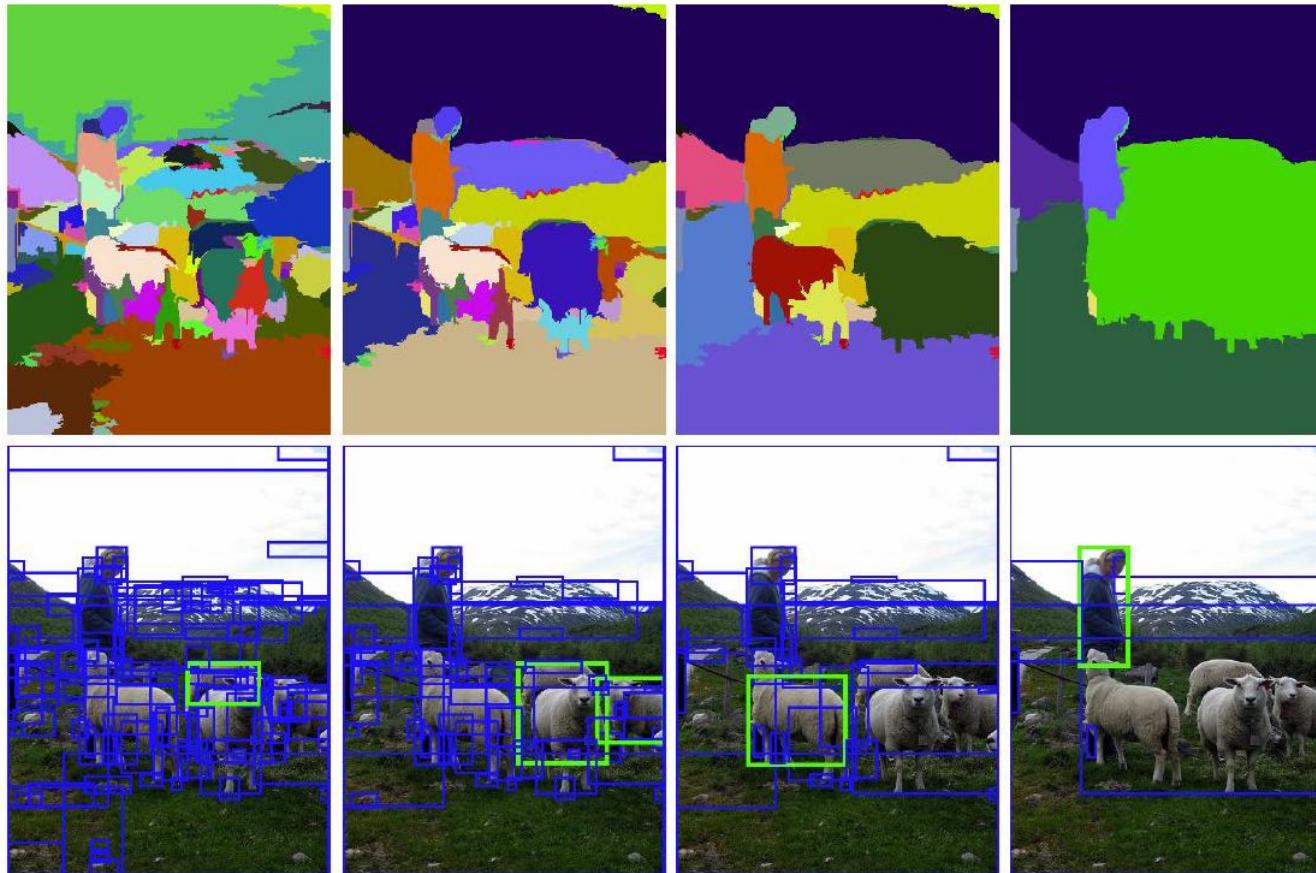


Selective Search: Method

- Take bounding boxes of all generated regions and treat them as possible object locations.



Selective Search: Method

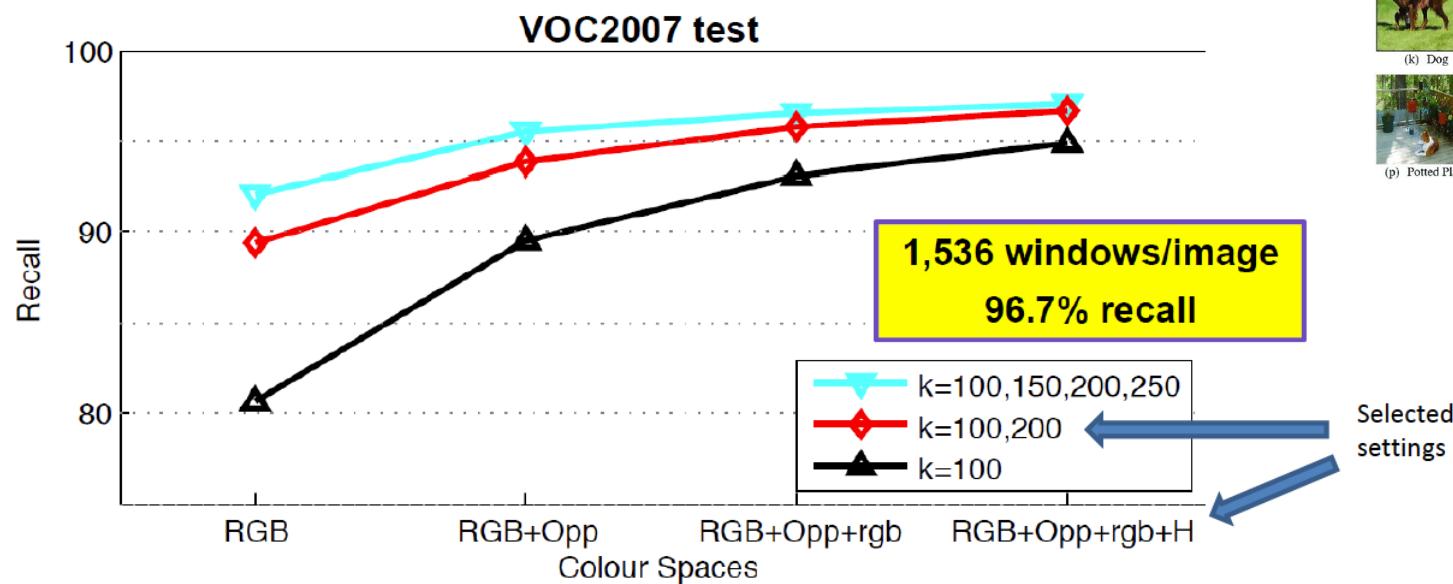


Selective Search: High Recall Revisited

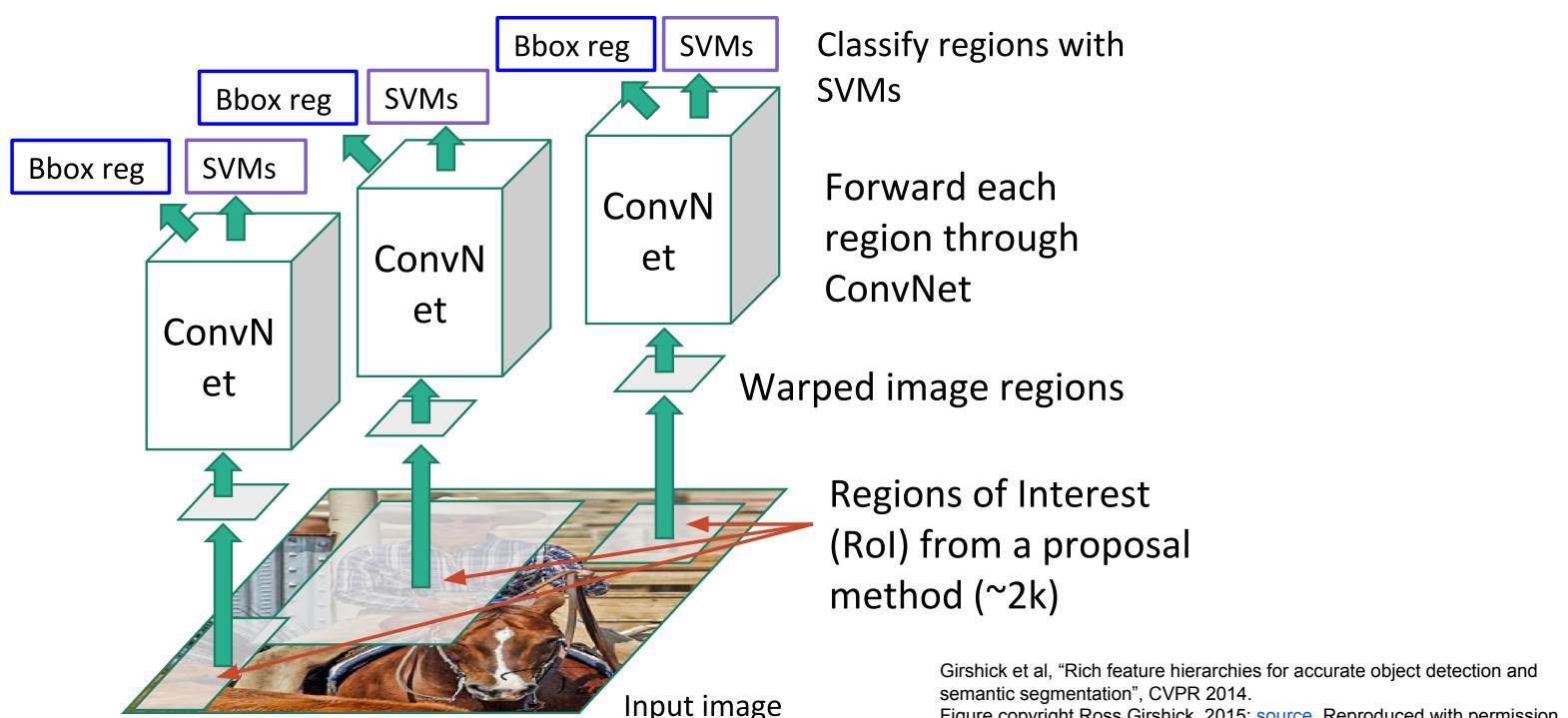
- No single segmentation works for all cases
 - > diversify the set of segmentations
- Use different color spaces
 - RGB, Opponent color, normalized RGB, and hue
- Use different parameters in Felzenswalb method
 - $k = [100, 150, 200, 250]$ (k = threshold parameter)

Selective Search: Evaluation of Object Hypotheses

- Recall is a proportion of objects that are covered by some box with >0.5 overlap

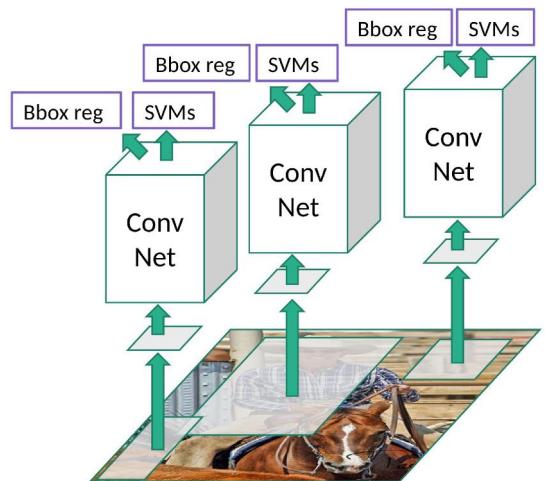


R-CNN — Region Based CNN



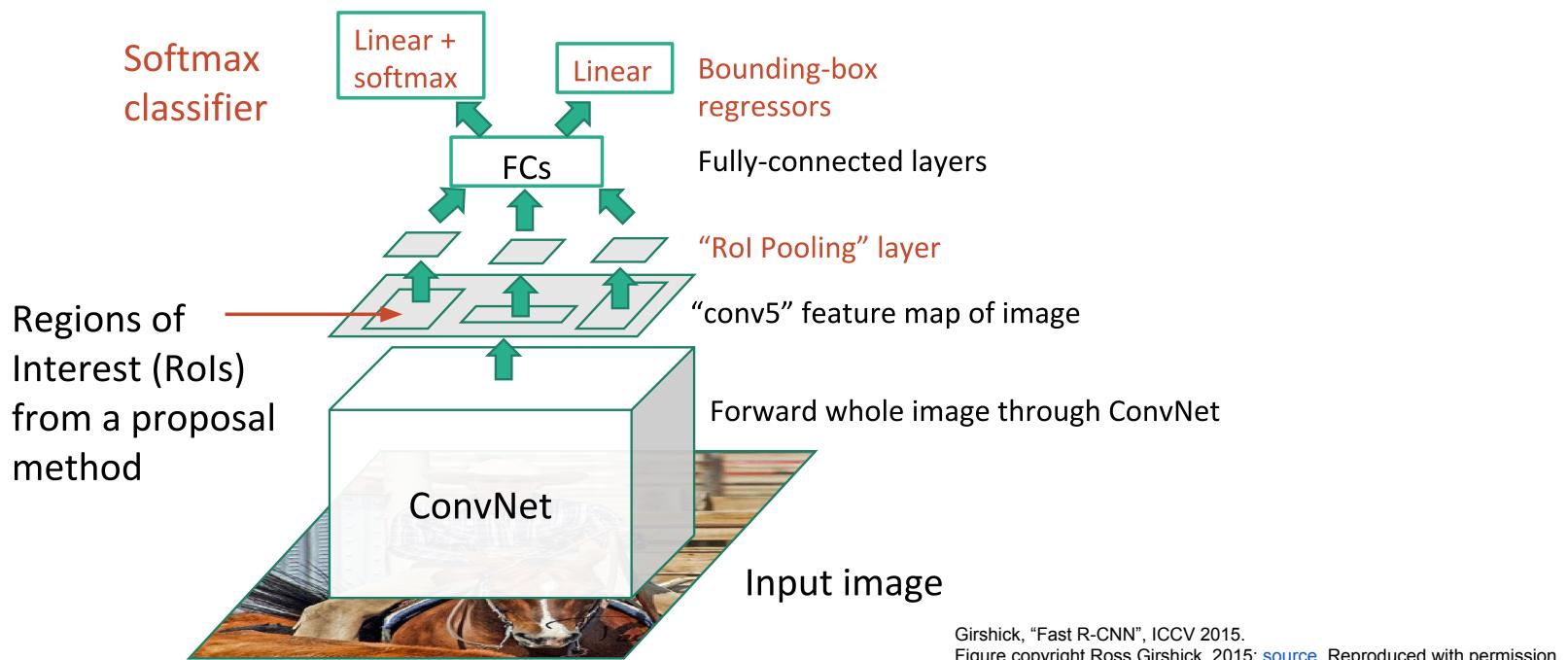
R-CNN — Region Based CNN: Problems

- 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]

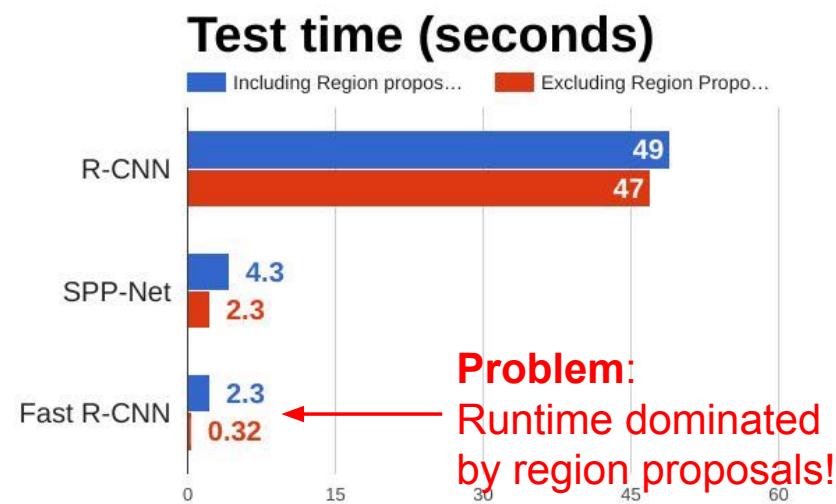
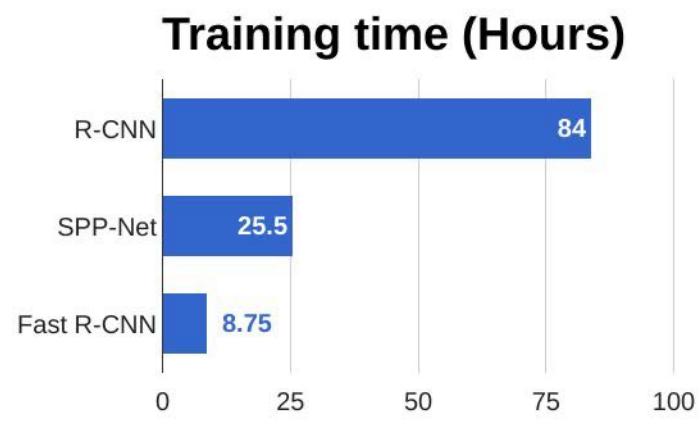


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014.
Slide copyright Ross Girshick, 2015; [source](#). Reproduced with permission.

Fast R-CNN



R-CNN vs. SPP vs. Fast-RCNN



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

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

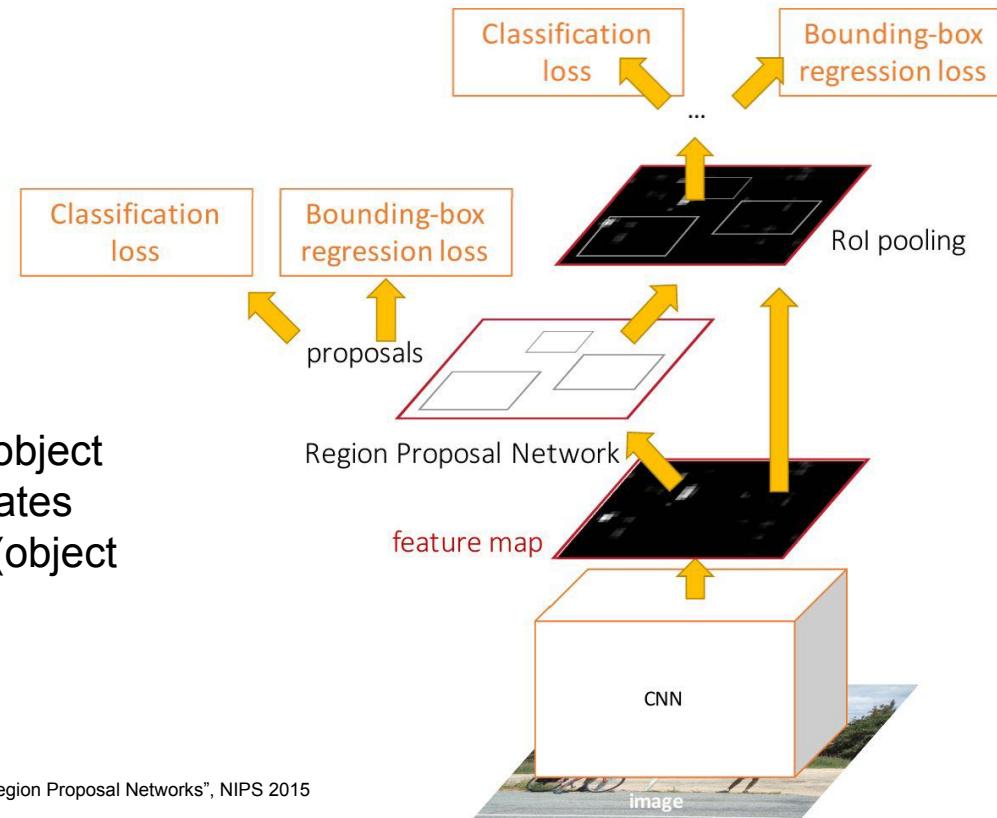


Faster - R-CNN: Make CNN do Proposals also !

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



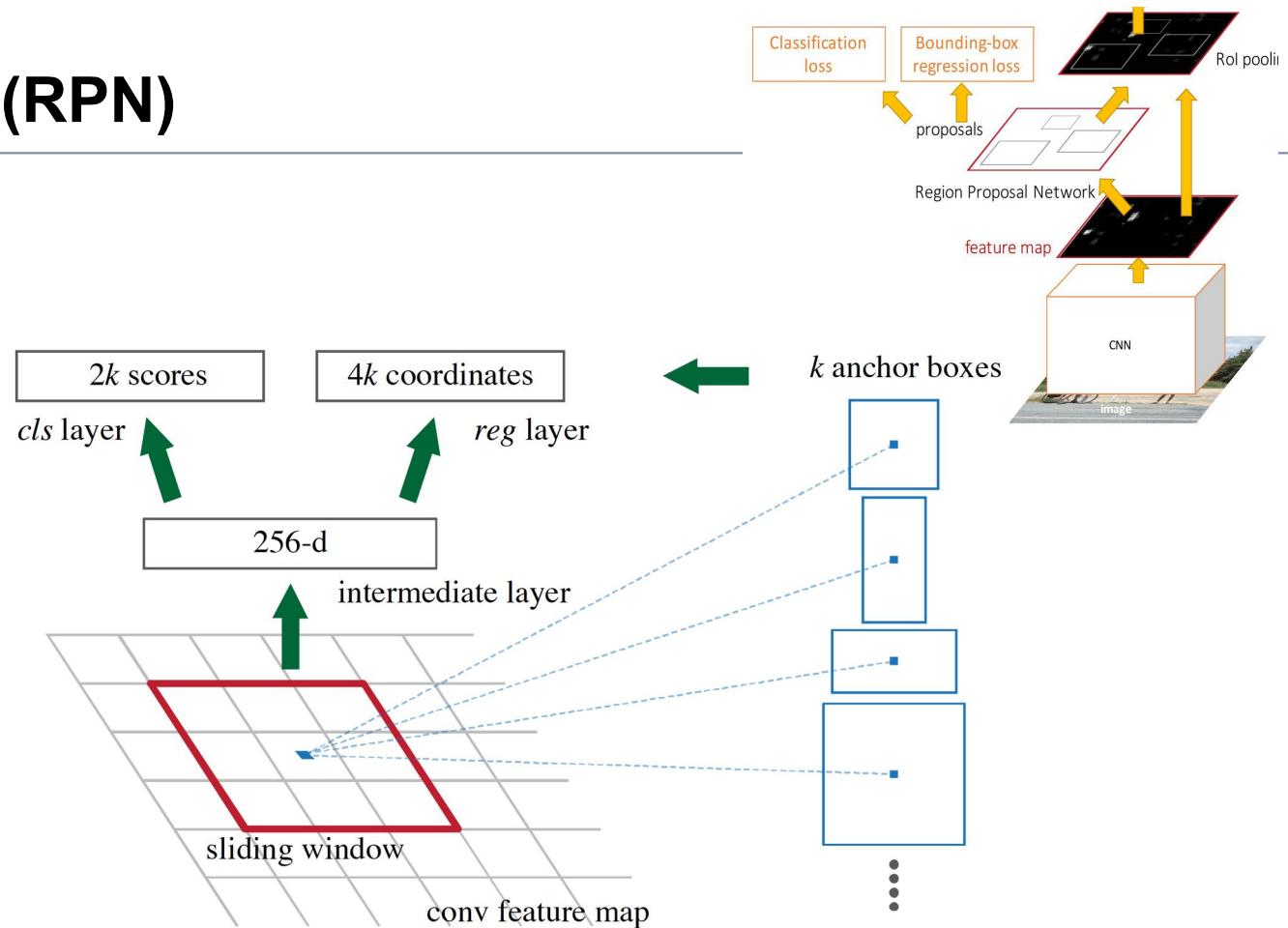
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

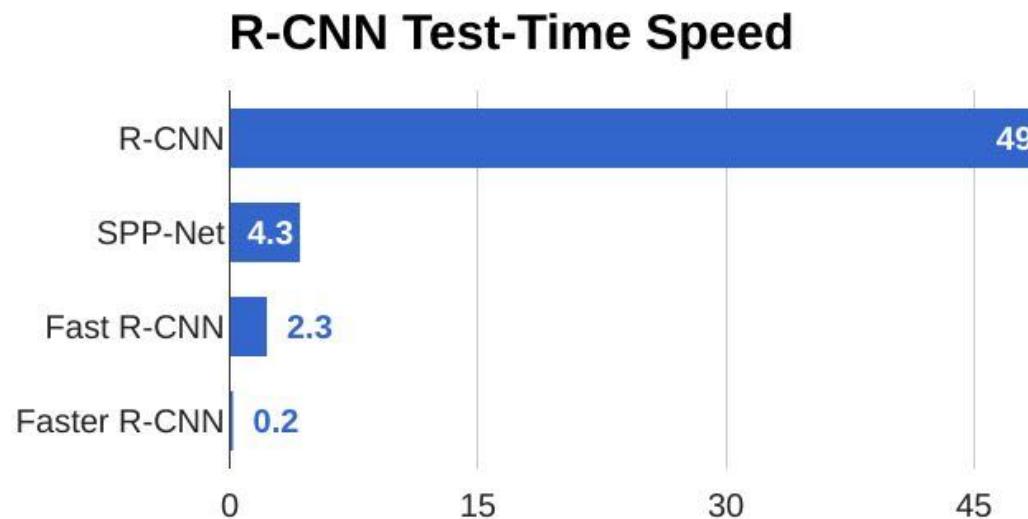


Region Proposal Network (RPN)

- components
 - ▶ 3x3 sliding window
 - ▶ 256-dimensional vector for each location (convolutions)
 - ▶ k anchor boxes, e.g. $k=9$:
 - 3 scales x 3 aspect ratios
 - ▶ for each box
 - class score (here 2-class softmax) for (any) object present or not
 - 4 coordinates for bounding box



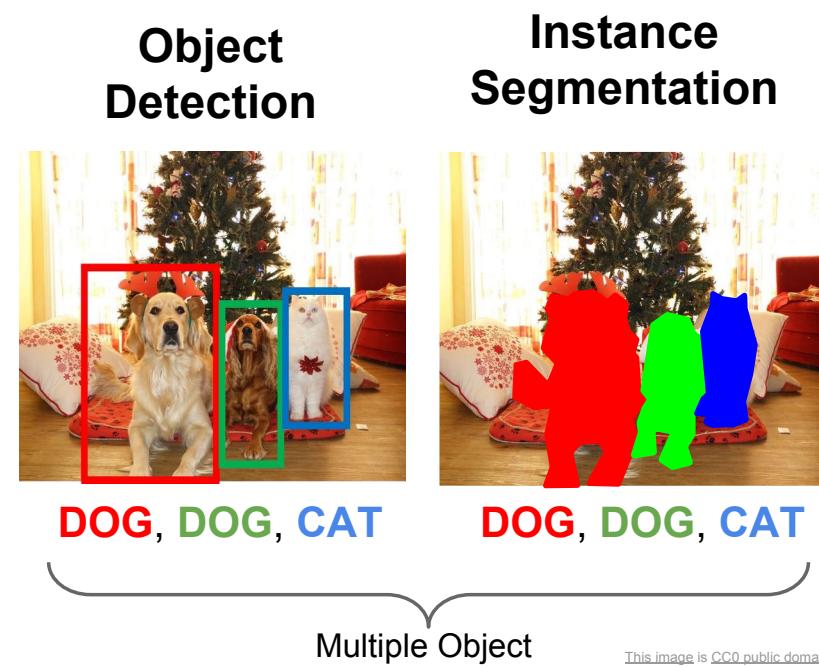
Faster - R-CNN



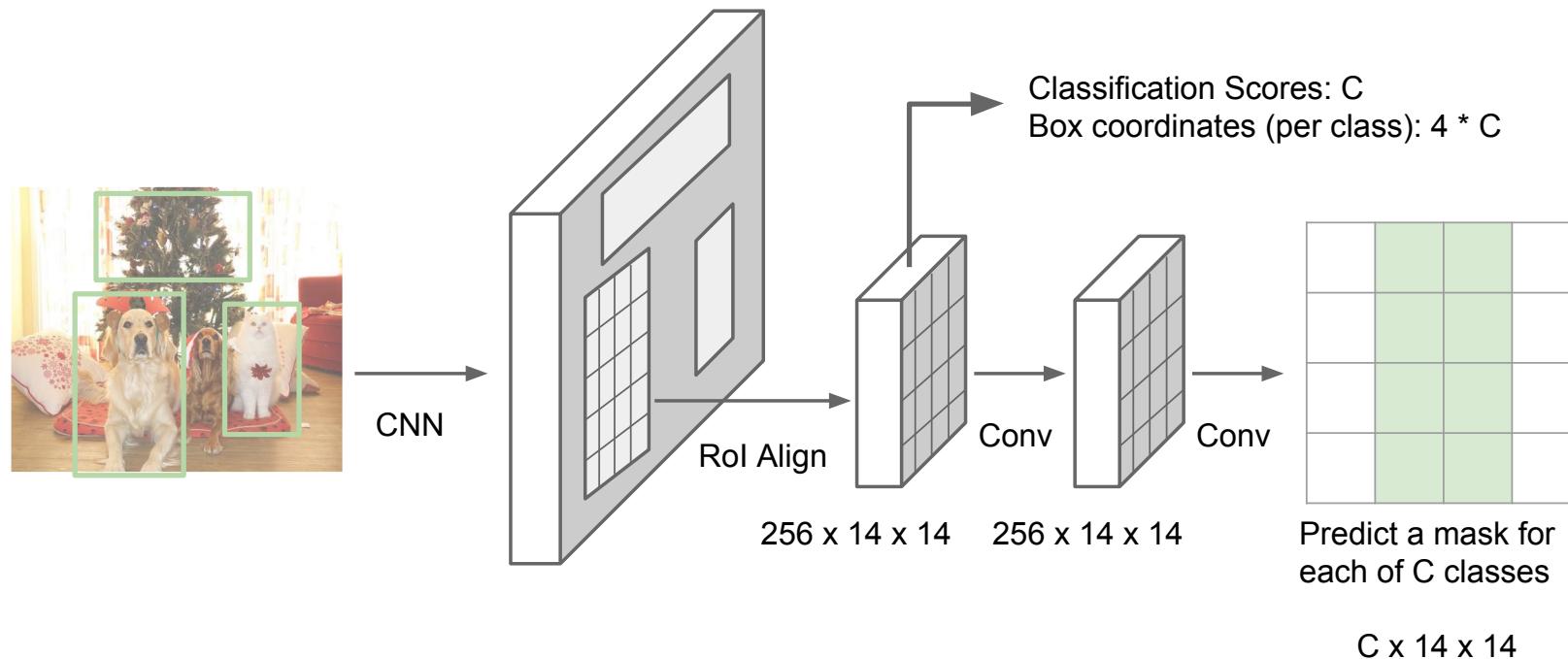
slide credit: Fei-Fei, Justin Johnson, Serena Yeung



Object Detection vs. Instance Segmentation



Mask R-CNN = Faster R-CNN + Segmentation Output for each ROI

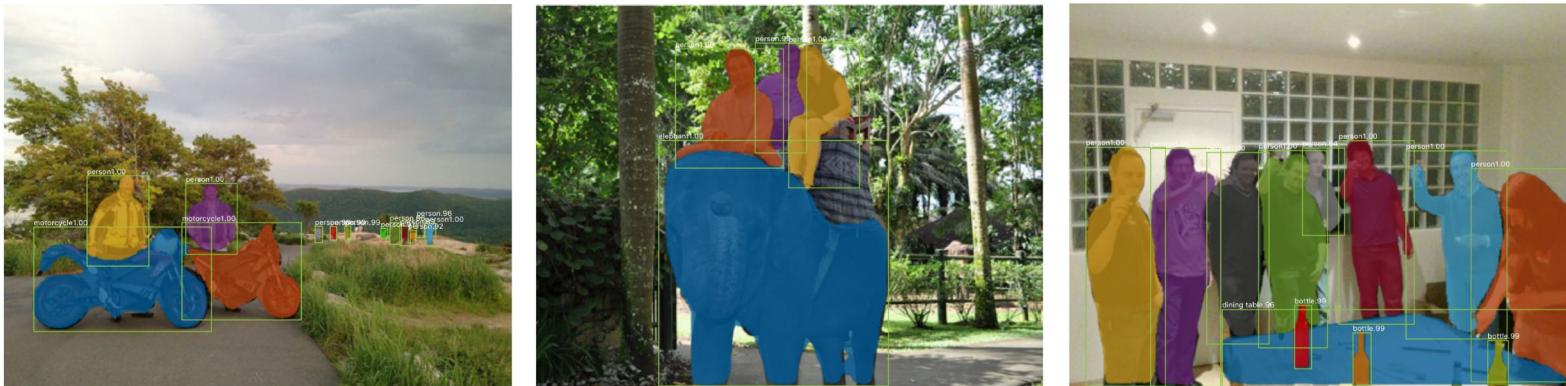


He et al, "Mask R-CNN", arXiv 2017

slide credit: Fei-Fei, Justin Johnson, Serena Yeung



Mask R-CNN: Very Good Results



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

Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung



Mask R-CNN: Also does Pose

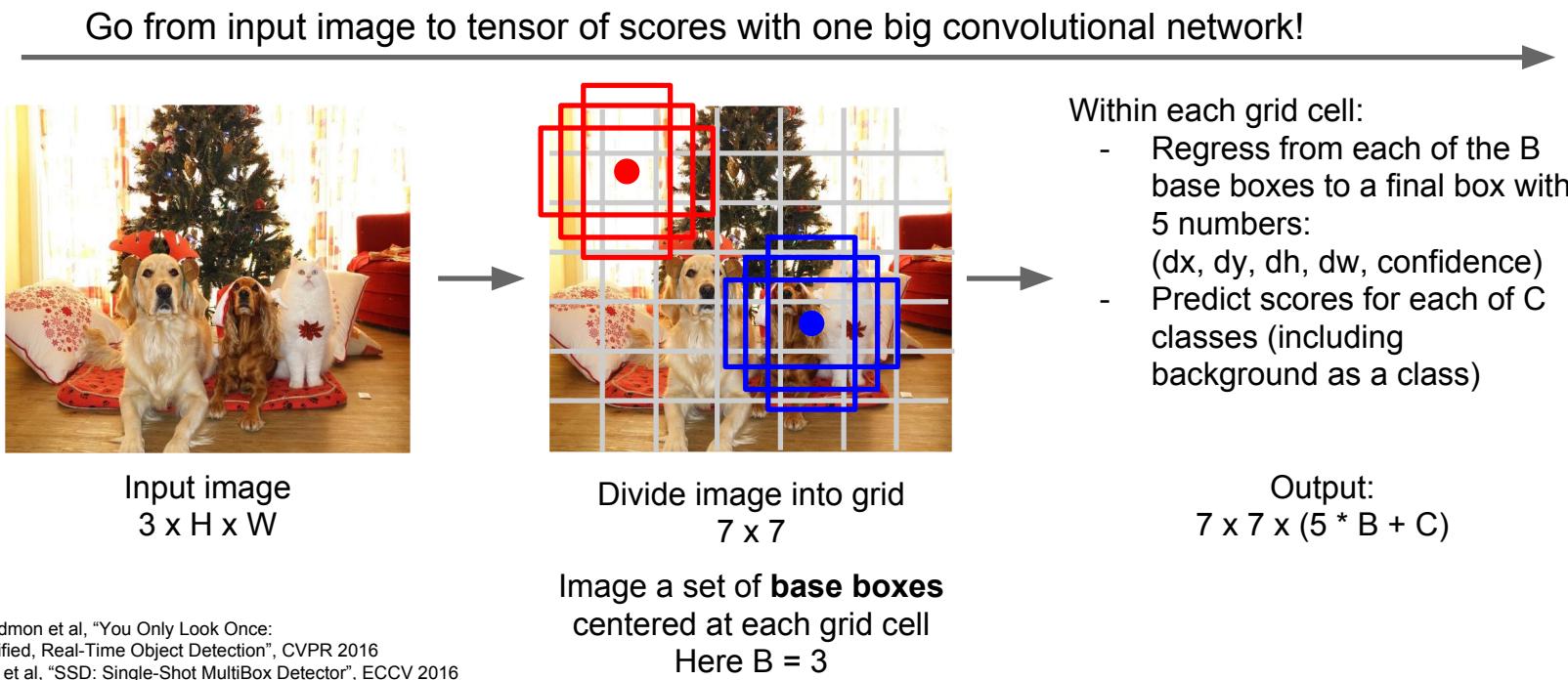


He et al, "Mask R-CNN", arXiv 2017

Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

slide credit: Fei-Fei, Justin Johnson, Serena Yeung

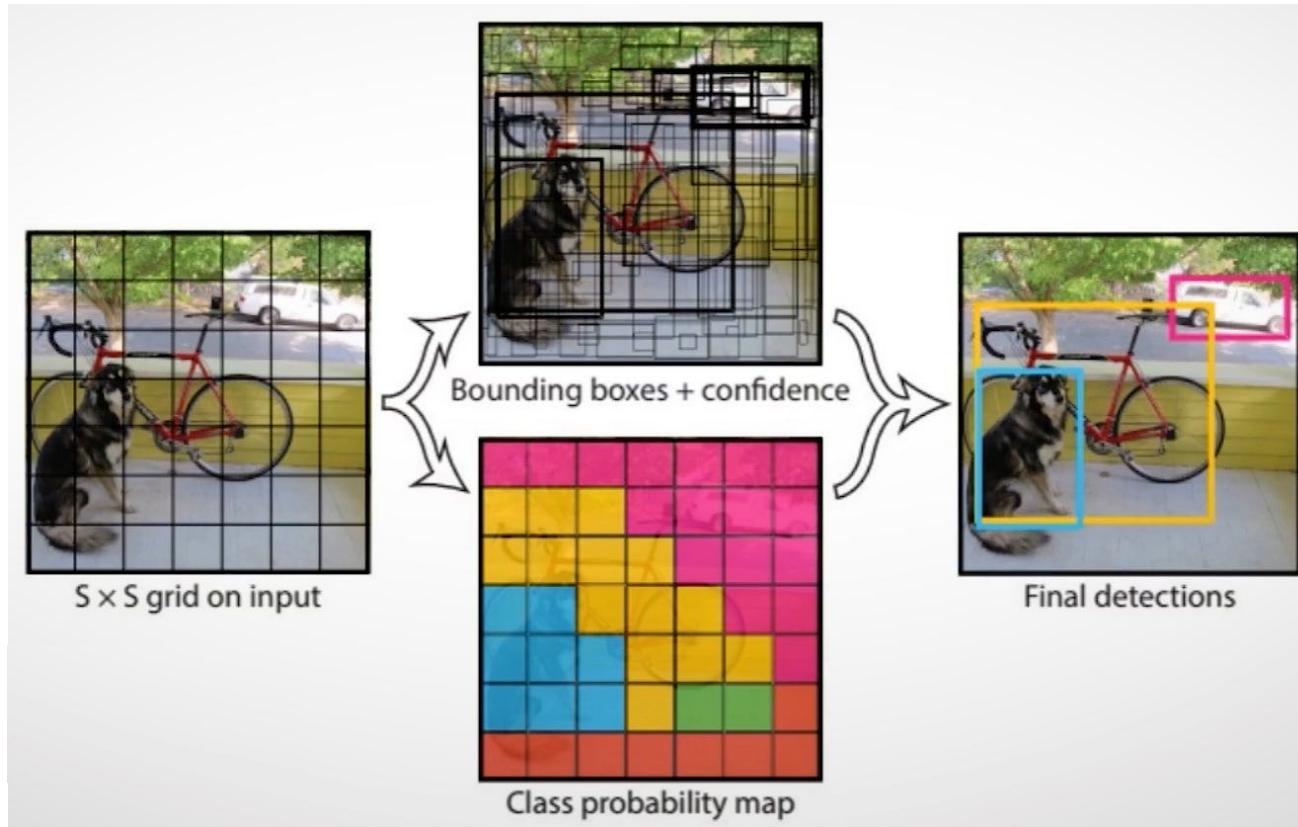
Detection without Proposals: YOLO (alternative: SSD)



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

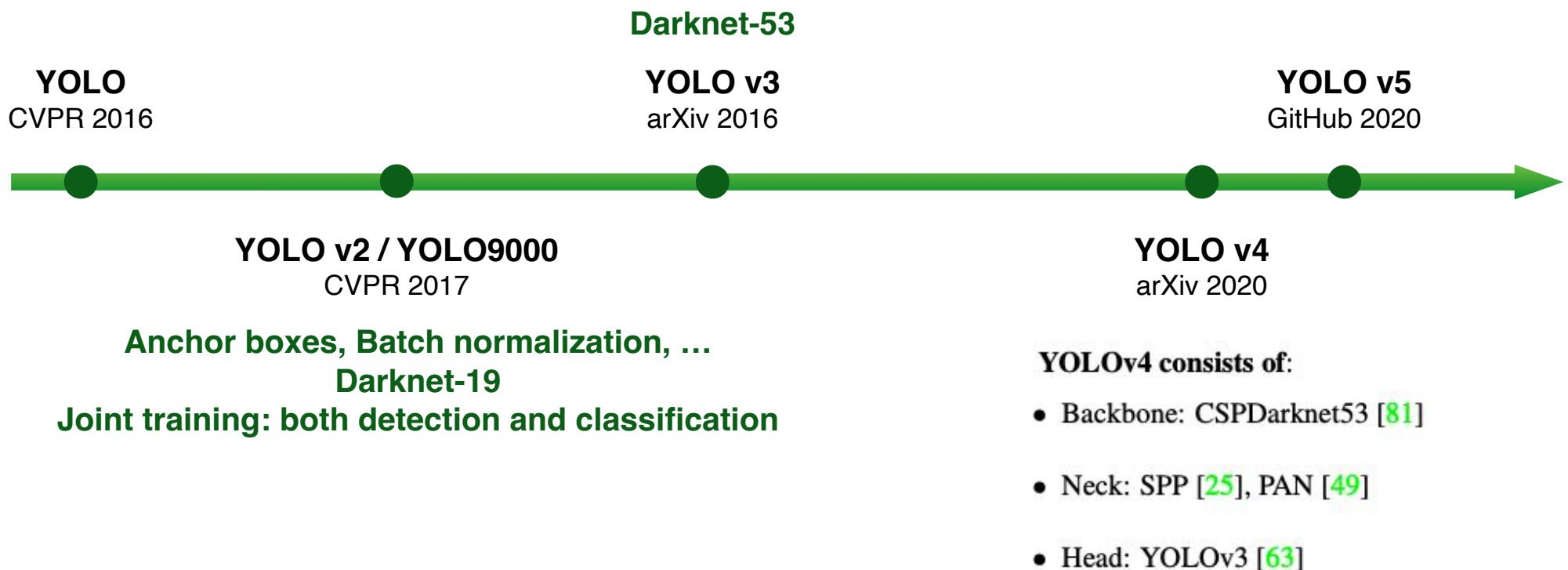
slide credit: Fei-Fei, Justin Johnson, Serena Yeung

Detection without Proposals: YOLO



slide credit: <https://youtu.be/YmMZkCstui0>

YOLO Family



Object Detection: Impact of Deep Learning

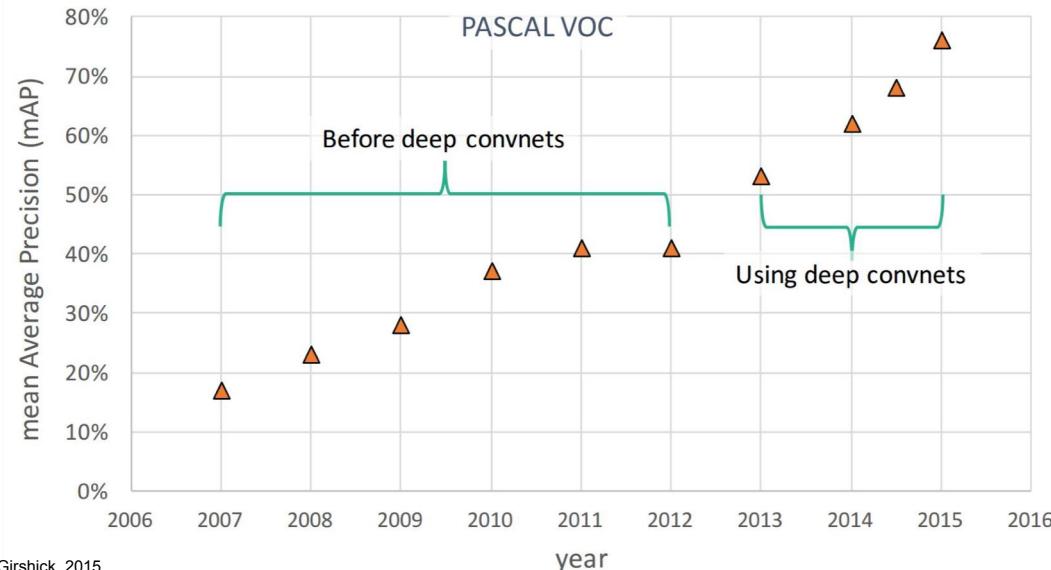


Figure copyright Ross Girshick, 2015.
Reproduced with permission.



slide credit: Fei-Fei, Justin Johnson, Serena Yeung