

*CS 1674: Intro to Computer Vision*

# Object Recognition

Prof. Adriana Kovashka  
University of Pittsburgh  
October 29, 2020

# Different Flavors of Object Recognition



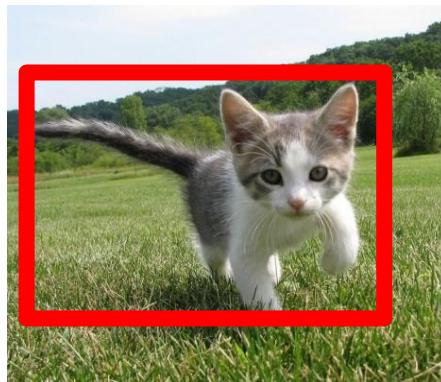
## Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

## Classification + Localization



CAT

Single Object

## Object Detection



DOG, DOG, CAT

Multiple Object

## Instance Segmentation

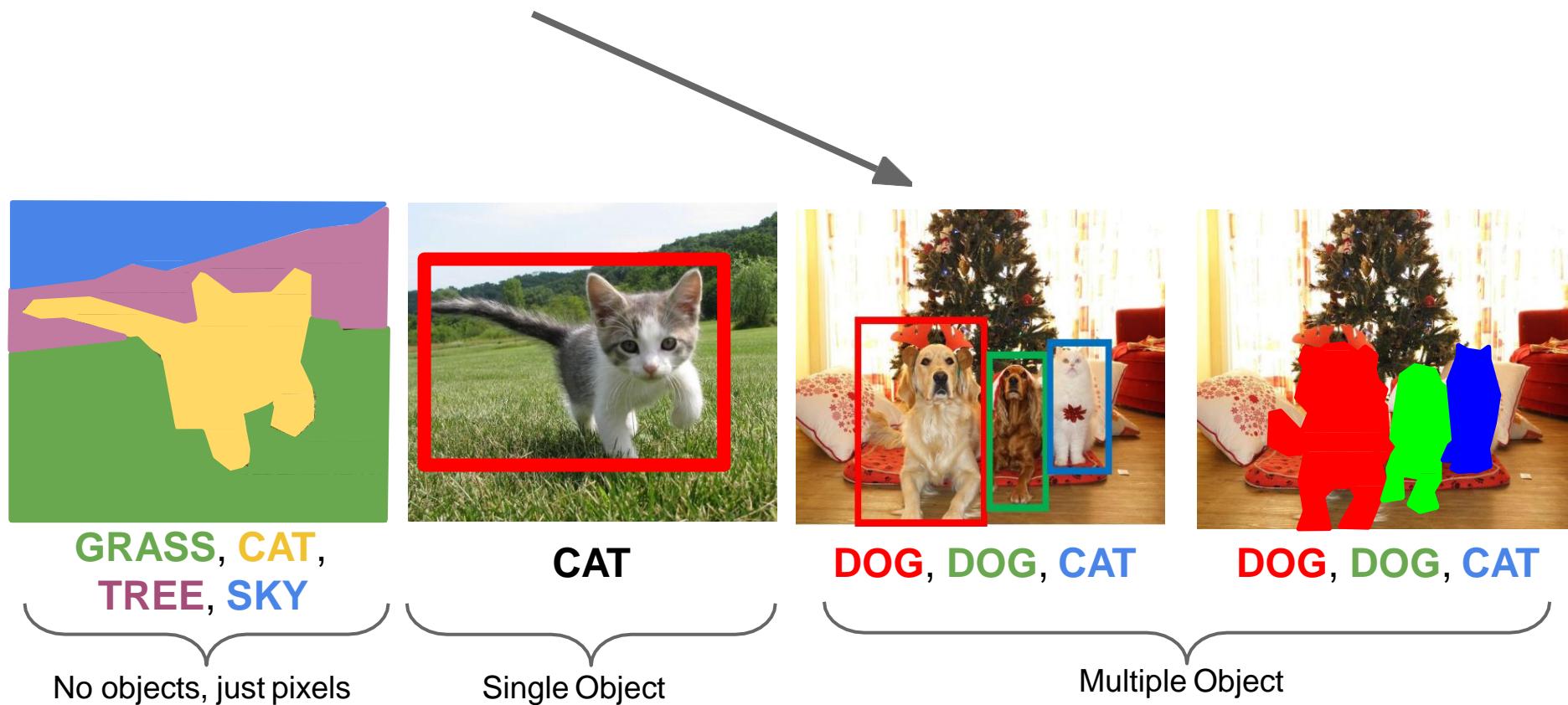


DOG, DOG, CAT

# Plan for the next three lectures

- Detection approaches
  - Pre-CNNs
    - Detection with whole windows: Pedestrian detection
    - Part-based detection: Deformable Part Models
  - Post-CNNs
    - Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
    - Detection without region proposals: YOLO, SSD
- Segmentation approaches
  - Semantic segmentation: FCN
  - Instance segmentation: Mask R-CNN

# Object Detection



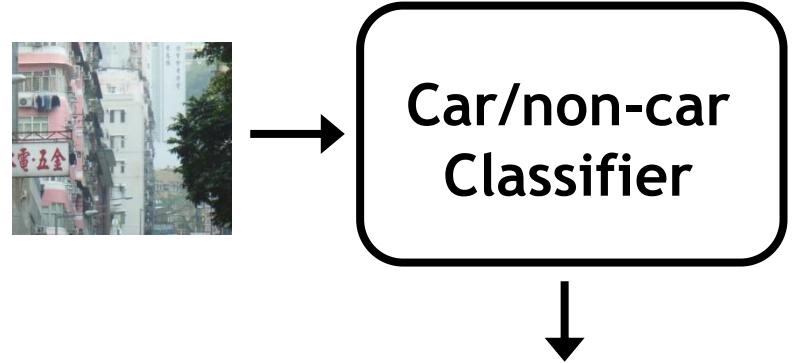
# Object detection: basic framework

- Build/train object model
- Generate candidate regions in new image
- Score the candidates

# Window-template-based models

## Building an object model

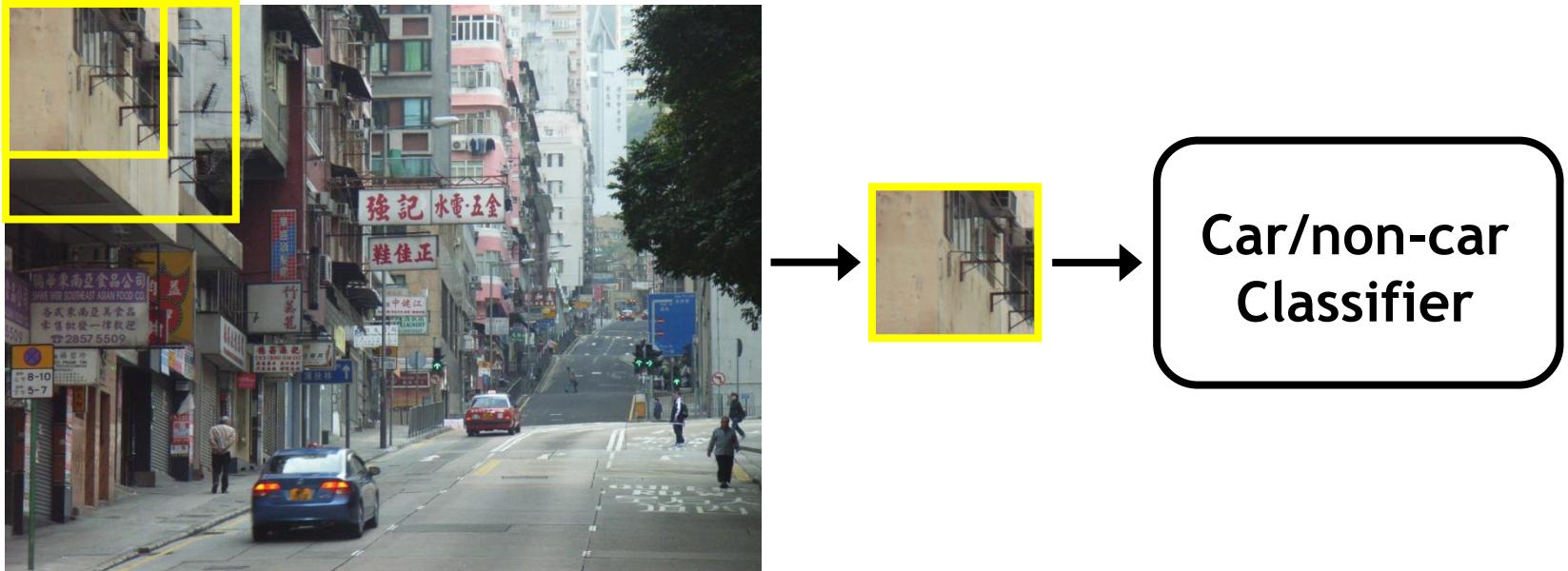
Given the representation, train a binary classifier



No, ~~yes~~, not car.

# Window-template-based models

## Generating and scoring candidates



Car/non-car  
Classifier

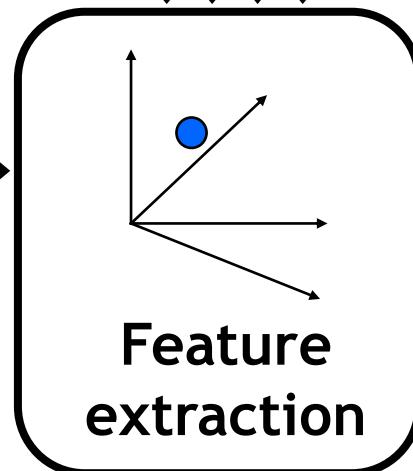
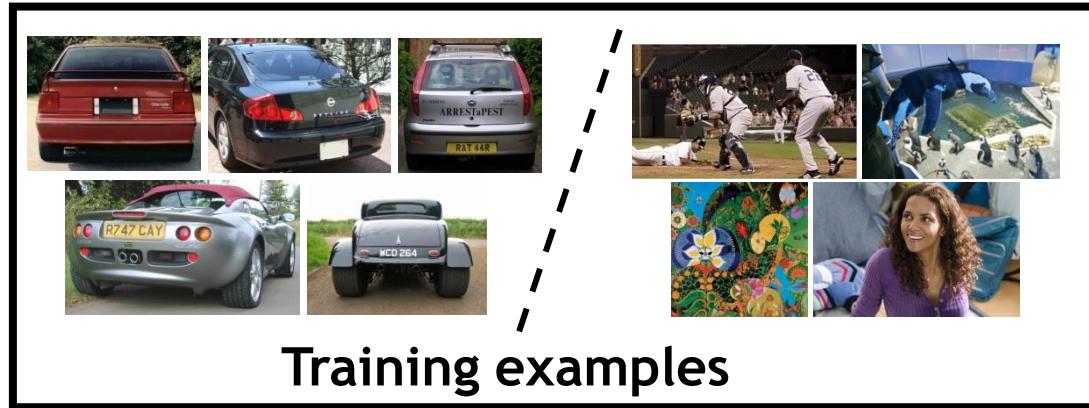
# Window-template-based object detection: recap

## Training:

1. Obtain training data
2. Define features
3. Define classifier

## Given new image:

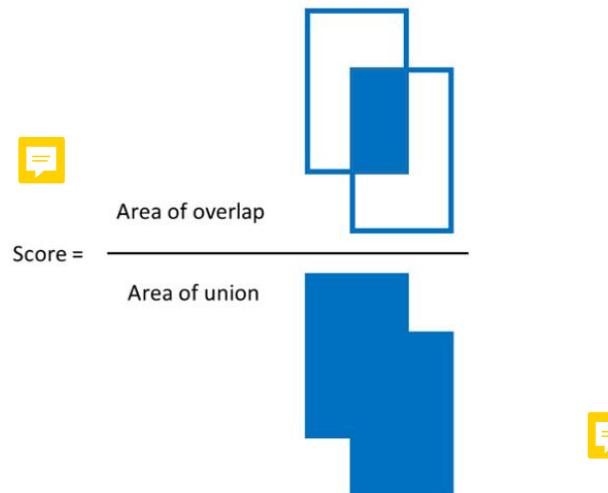
1. Slide window
2. Score by classifier



# Evaluating detection methods

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{\#TP(c)}{\#TP(c) + \#FP(c)}$$

- True Positive - TP(c): a predicted bounding box (pred\_bb) was made for class c, there is a ground truth bounding box (gt\_bb) of class c, and  $\text{IoU}(\text{pred\_bb}, \text{gt\_bb}) \geq 0.5$ .
- False Positive - FP(c): a pred\_bb was made for class c, and there is no gt\_bb of class c. Or there is a gt\_bb of class c, but  $\text{IoU}(\text{pred\_bb}, \text{gt\_bb}) < 0.5$ .



# Dalal-Triggs pedestrian detector

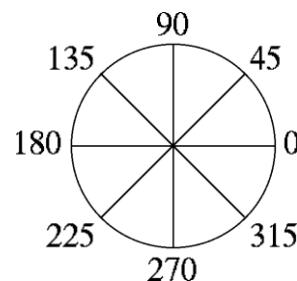


1. Extract fixed-sized (64x128 pixel) window at multiple positions and scales
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

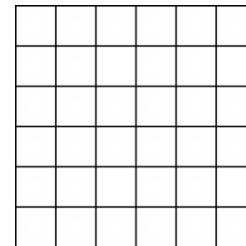
# Histograms of oriented gradients (HOG)

Divide image into 8x8 regions

Orientation: 9 bins  
(for unsigned angles)



Histograms in  
8x8 pixel cells



Votes weighted by magnitude



# Train SVM for pedestrian detection using HoG



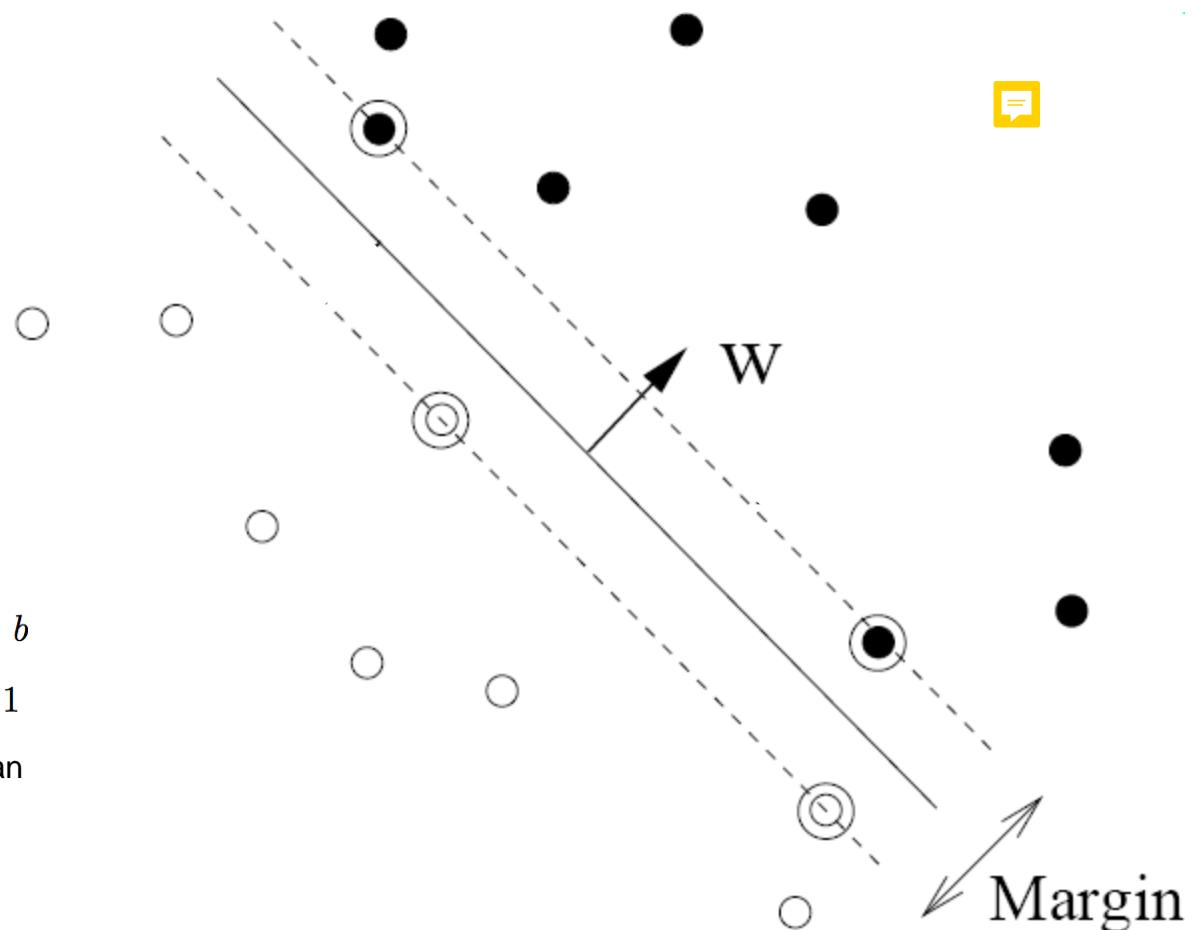
pos w



$$0.16 = w^T x + b$$

$$\text{sign}(0.16) = 1$$

$\Rightarrow$  pedestrian



# Remove overlapping detections

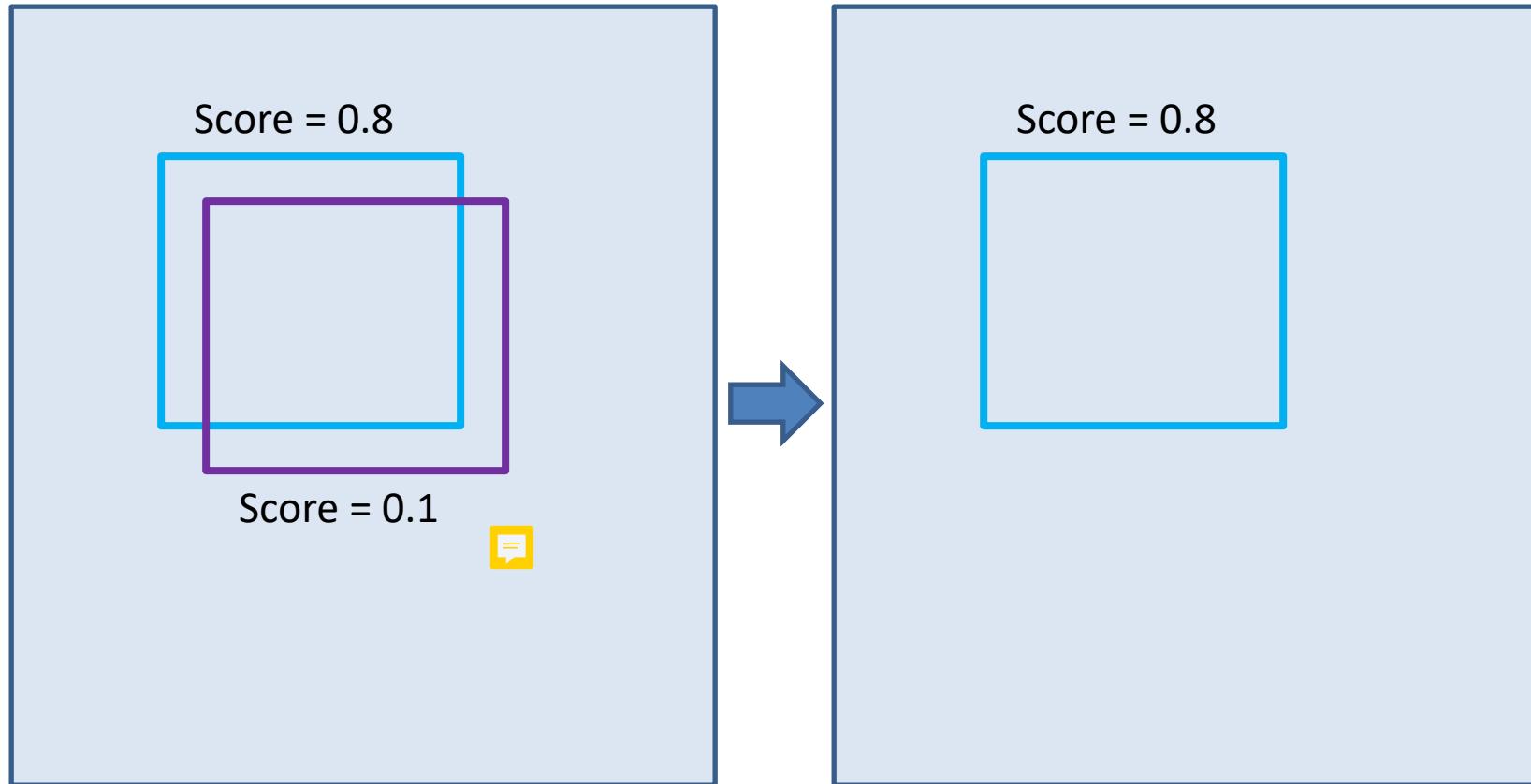


Area of overlap



Area of union

## Non-max suppression



# Are window templates enough?

- Many objects are articulated, or have parts that can vary in configuration

Images from Caltech-256, D. Ramanan



- Many object categories look very different from different viewpoints, or from instance to instance



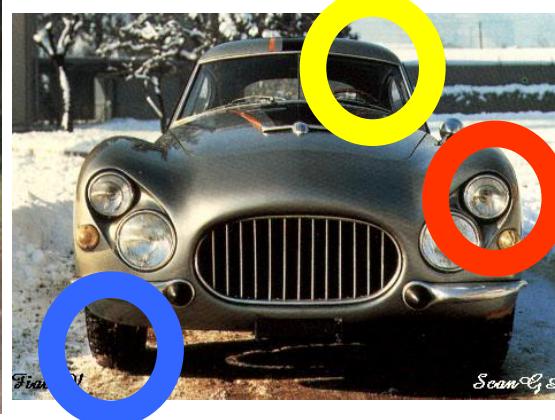
Adapted from N. Snavely, D. Tran

# Parts-based Models



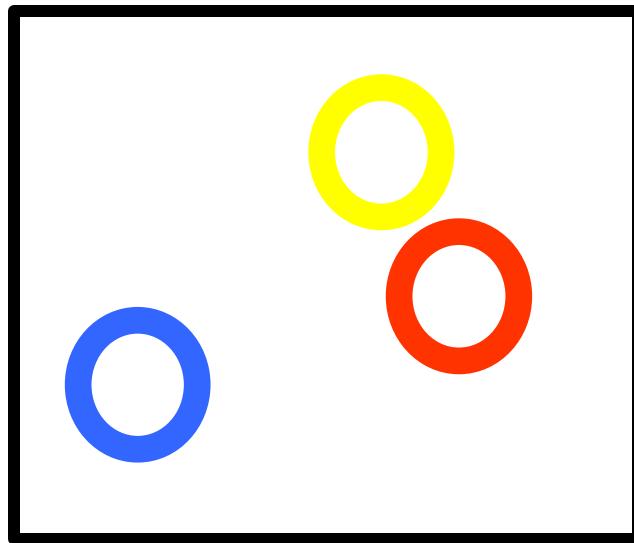
Define object by collection of parts modeled by

1. Appearance
2. Spatial configuration



# How to model spatial relations?

- One extreme: fixed template



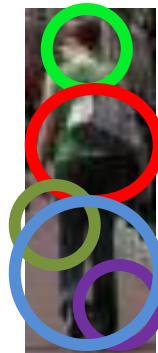
# Fixed part-based template

- Object model = sum of scores of features at fixed positions



$$+3 \text{ } +2 \text{ } -2 \text{ } -1 \text{ } -2.5 = -0.5 > 7.5$$

Non-object

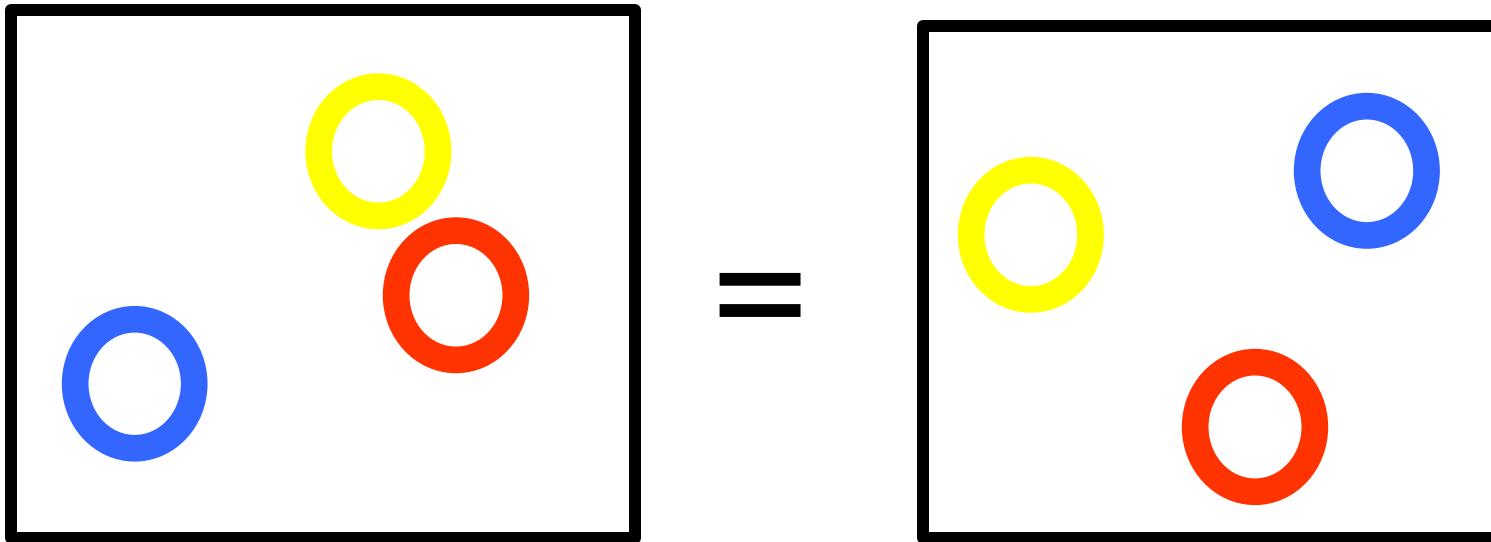


$$+4 \text{ } +1 \text{ } +0.5 \text{ } +3 \text{ } +0.5 = 10.5 > 7.5$$

Object

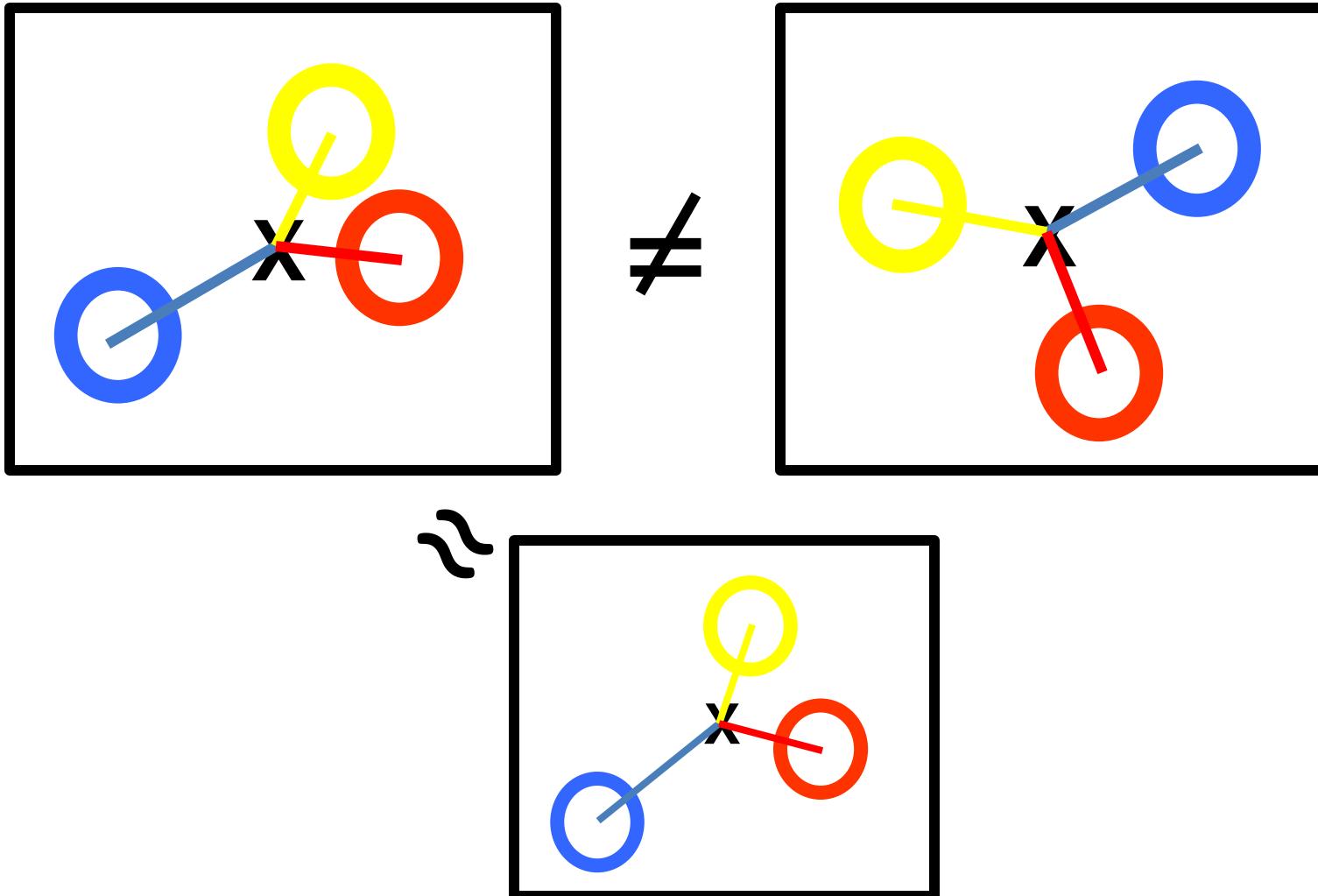
# How to model spatial relations?

- Another extreme: bag of words 



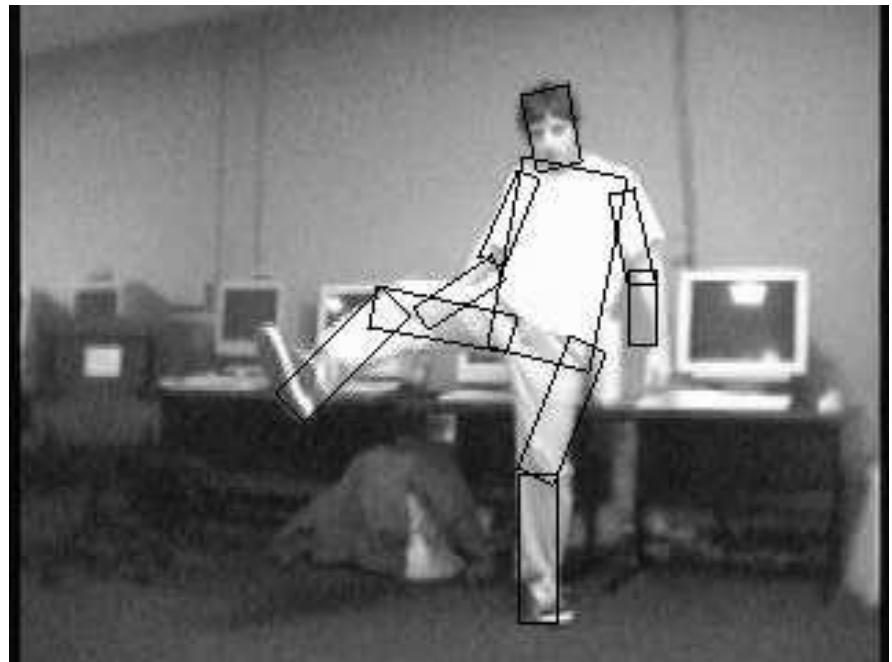
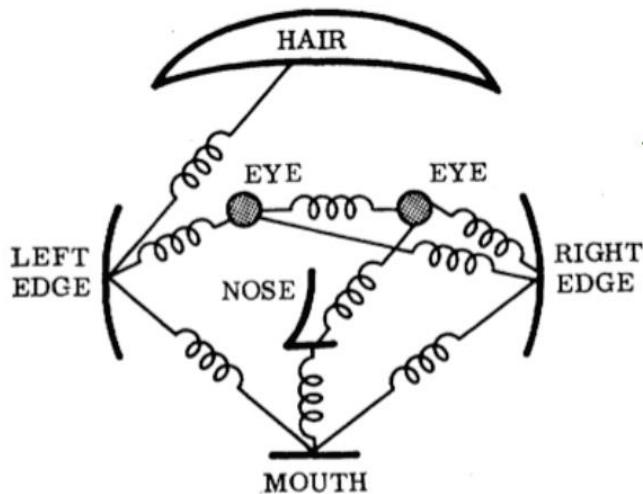
# How to model spatial relations?

- Star-shaped model 



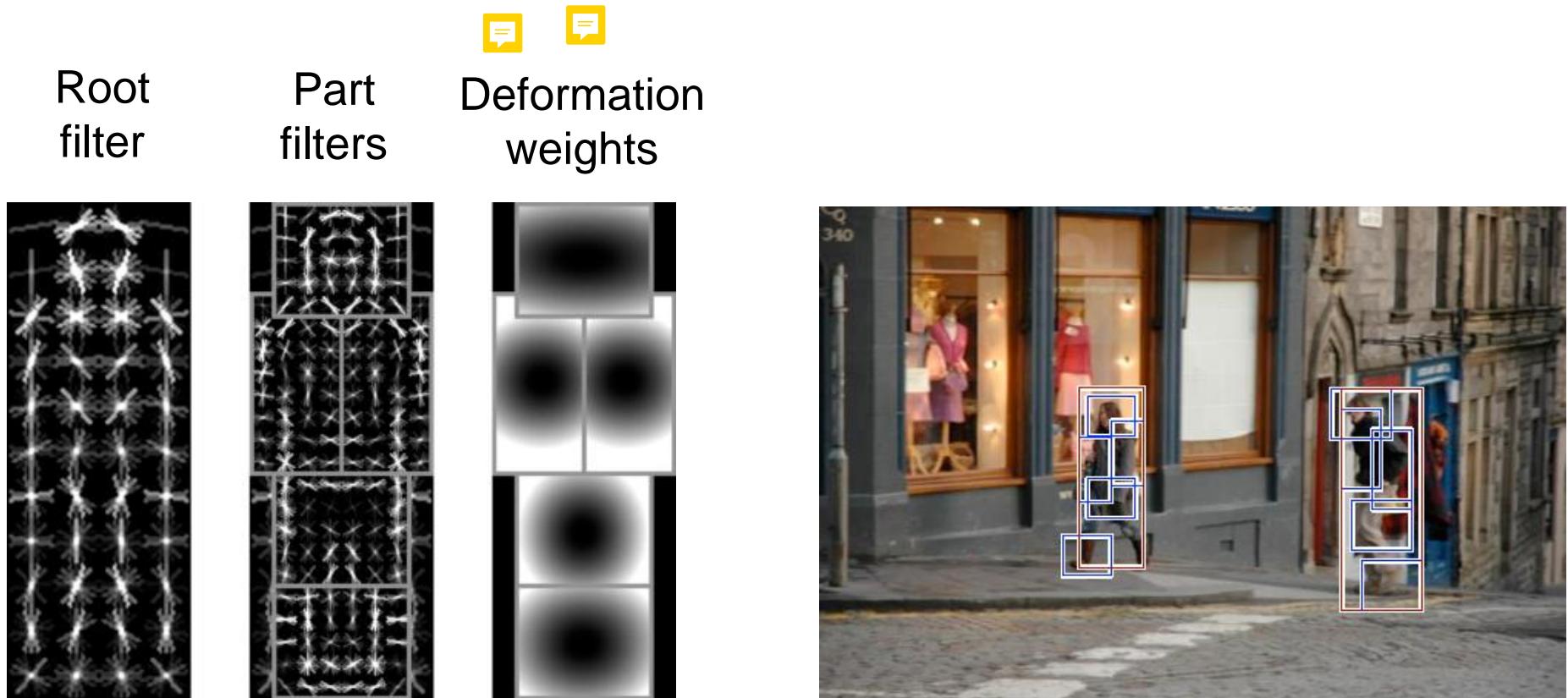
# Parts-based Models

- Articulated parts model
  - Object is configuration of parts
  - Each part is detectable and can move around



# Deformable Part Models

---



P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, [Object Detection with Discriminatively Trained Part Based Models](#), PAMI 32(9), 2010

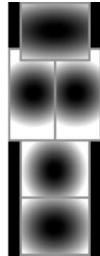
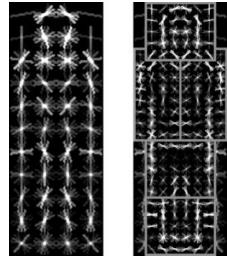
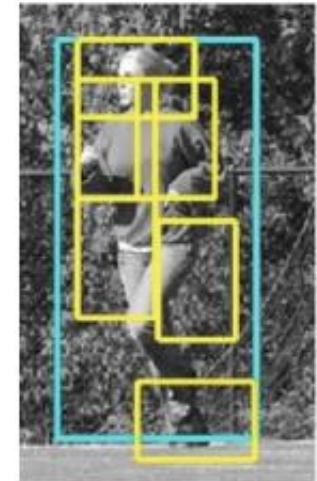
# Scoring an object hypothesis

- The score of a hypothesis is the sum of appearance scores minus the sum of deformation costs 

$$z = (p_0, \dots, p_n)$$

$p_0$ : location of root

$p_1, \dots, p_n$  : location of parts 



$$\text{score}(p_0, \dots, p_n) =$$

$$\sum_{i=0}^n F'_i \cdot \phi(H, p_i) - \sum_{i=1}^n d_i \cdot \phi_d(dx_i, dy_i) + b$$

Appearance weights

Part features

$$\begin{array}{ll} \text{part loc} & \text{anchor loc (where we} \\ (\text{where we see part}) & \text{expect to see part}) \\ (dx_i, dy_i) = (x_i, y_i) - (2(x_0, y_0) + v_i) \end{array}$$

## Displacements

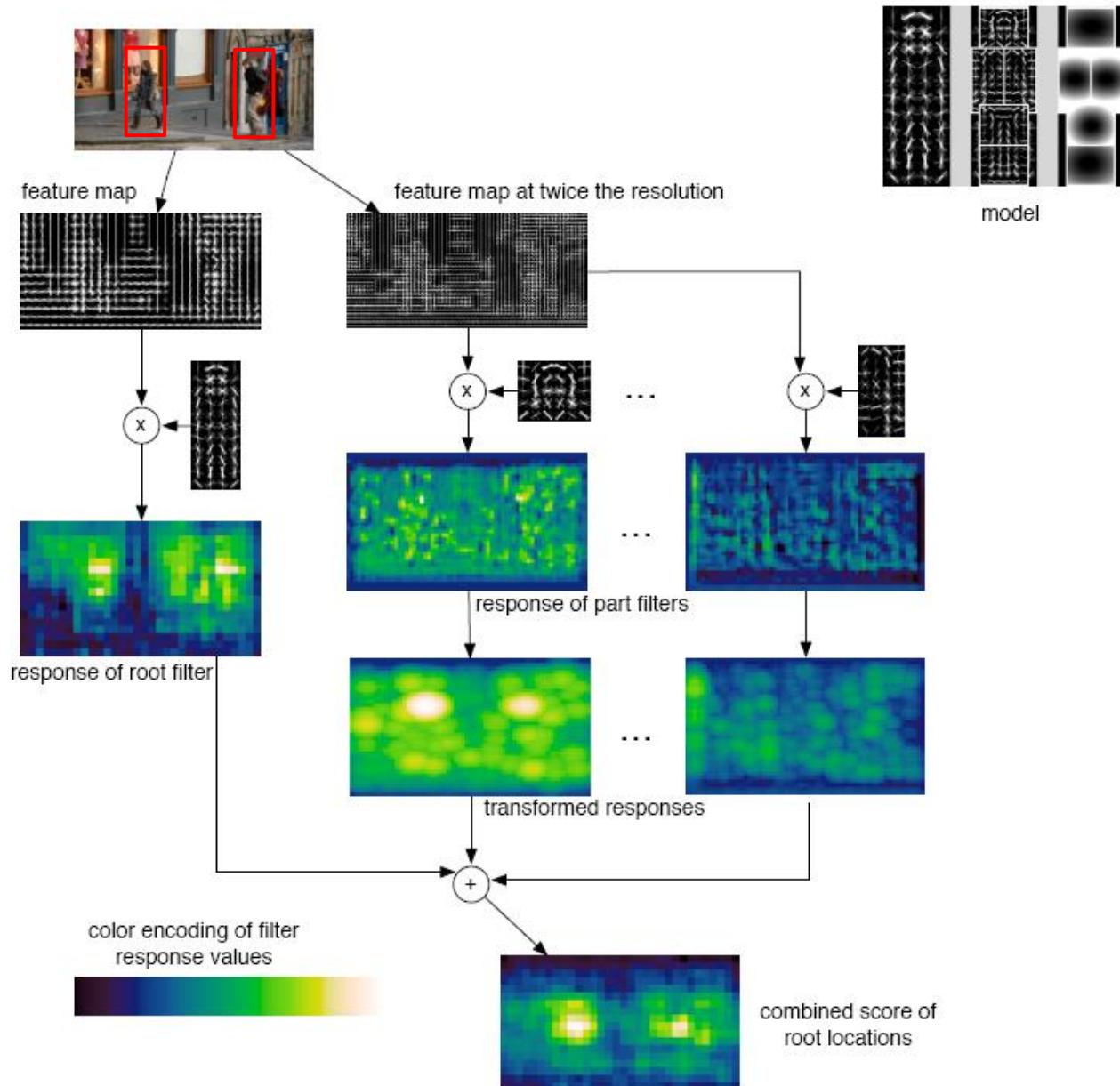
i.e. how much the part  $p_i$  moved from its expected anchor location in the x, y directions

## Deformation weights

i.e. how much we'll penalize the part  $p_i$  for moving from its expected location

# Detection

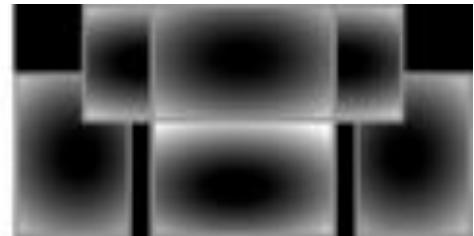
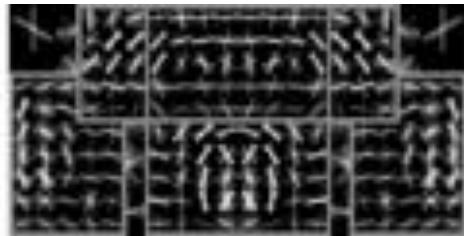
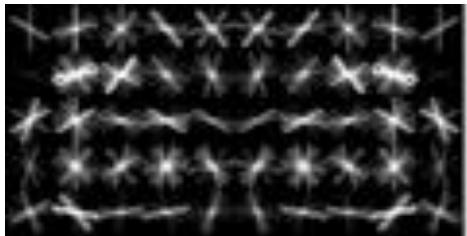
---



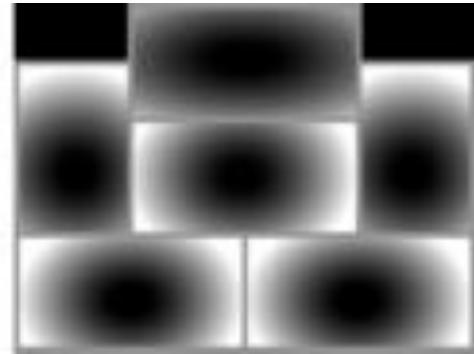
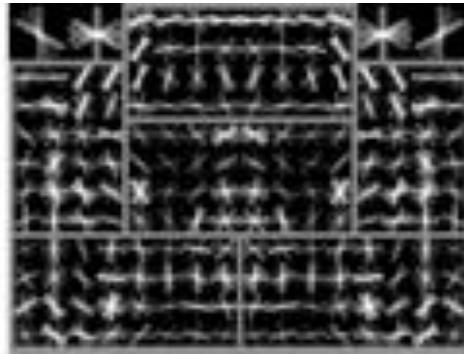
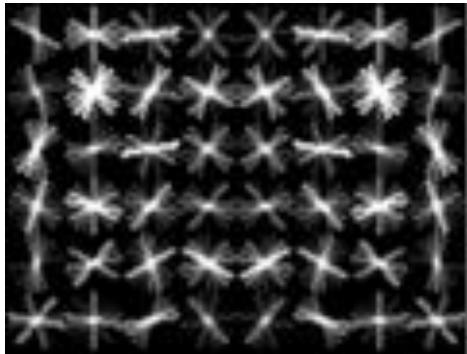
# Car model

---

Component 1



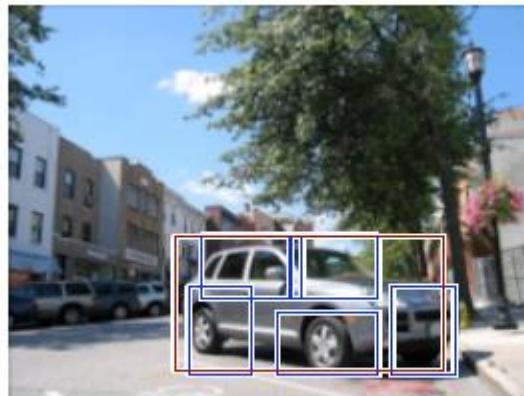
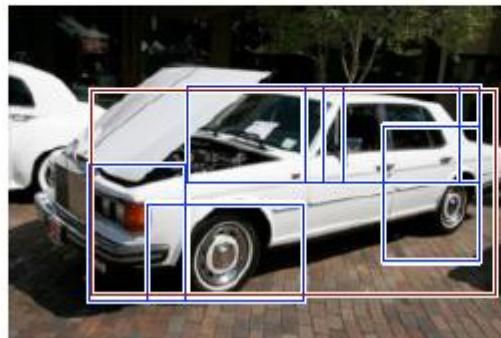
Component 2



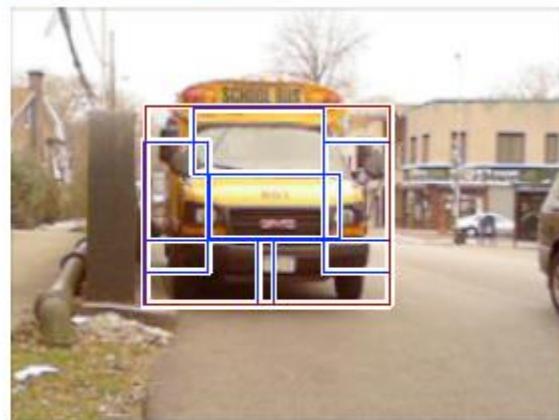
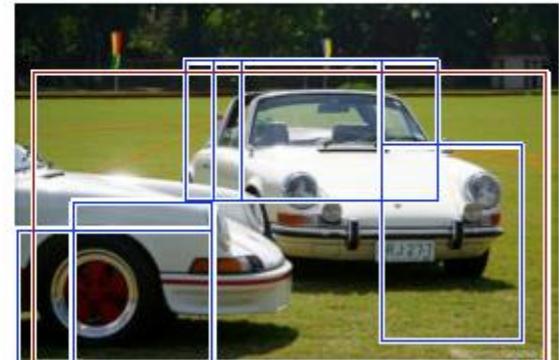
# Car detections

---

high scoring true positives

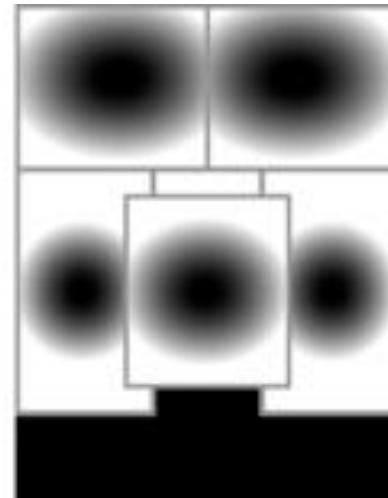
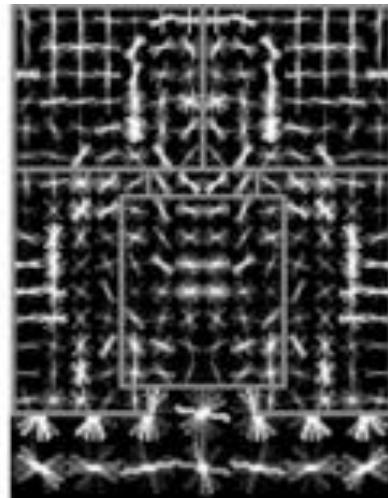
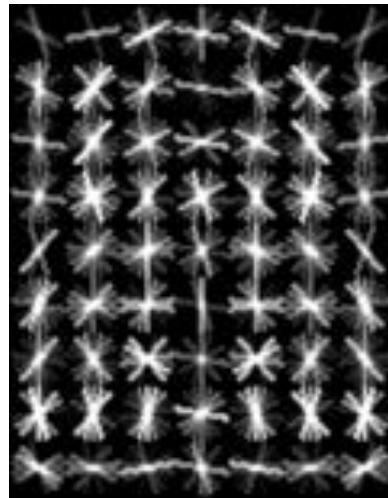
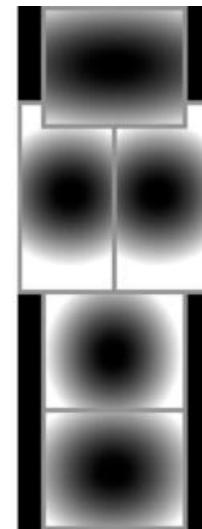
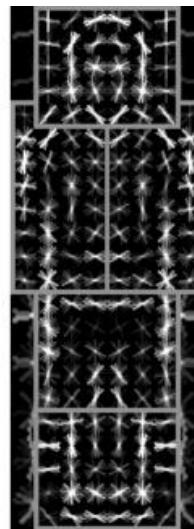
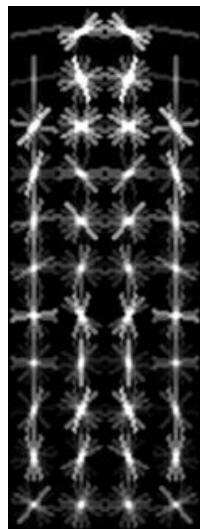


high scoring false positives



# Person model

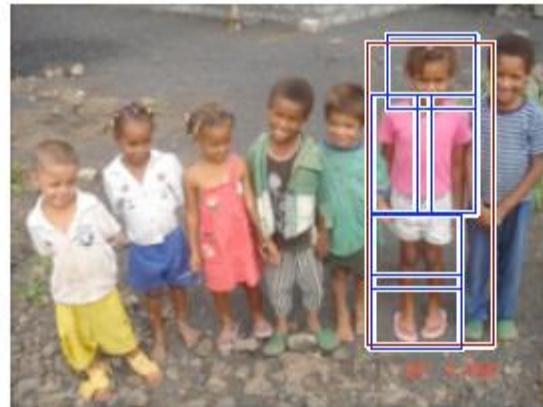
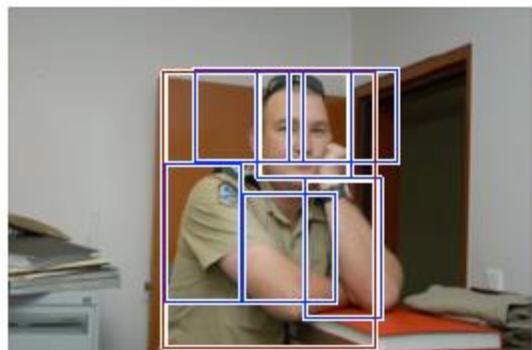
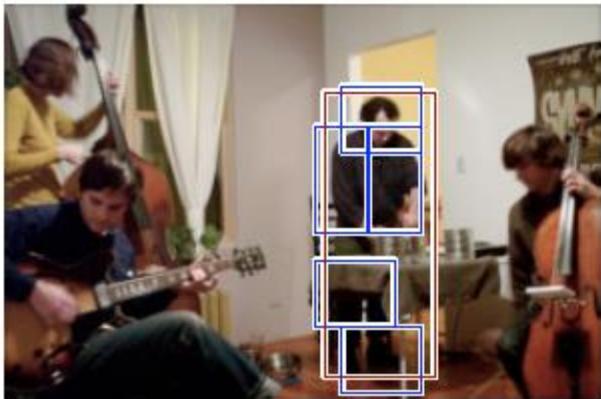
---



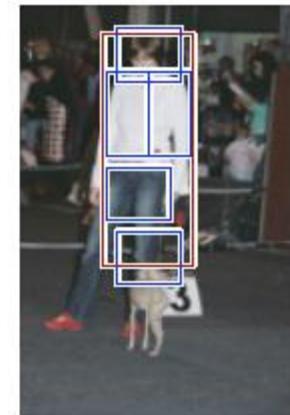
# Person detections

---

high scoring true positives

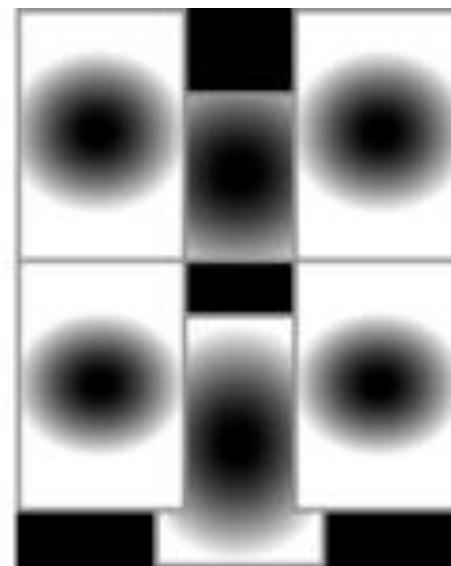
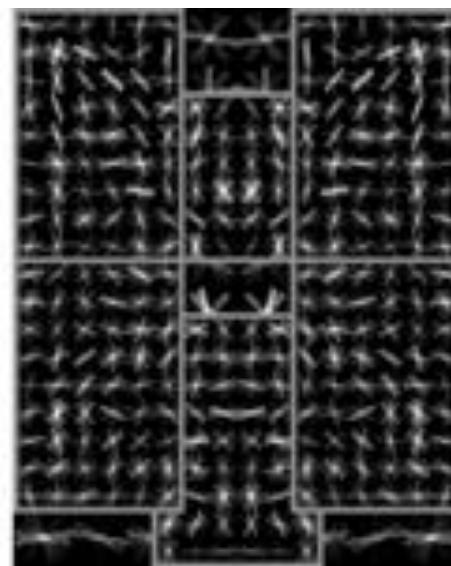
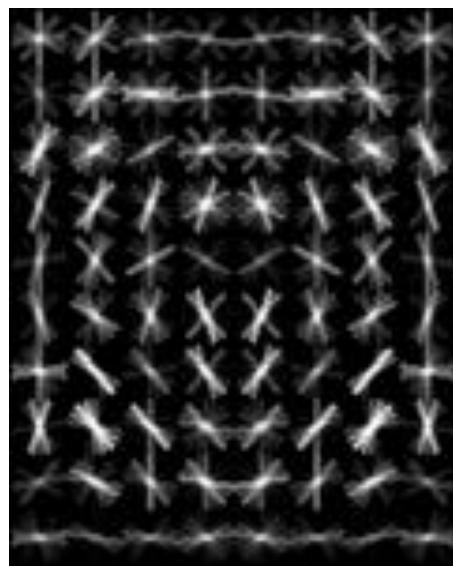
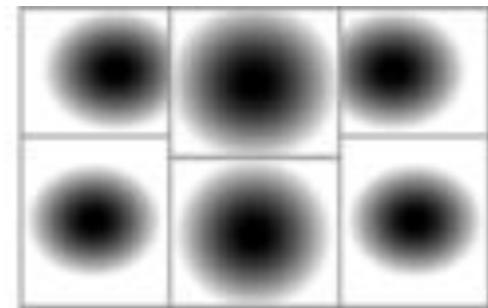
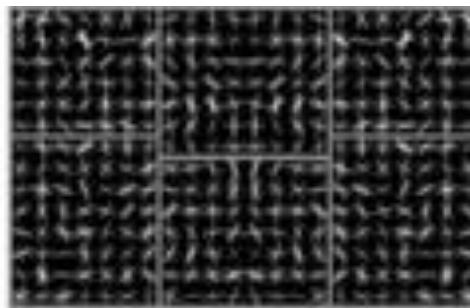
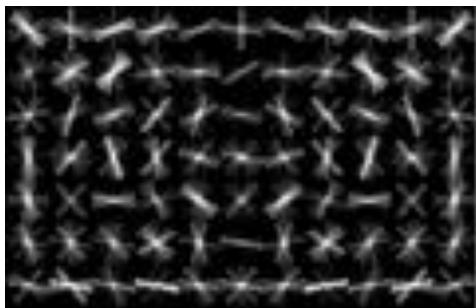


high scoring false positives  
(not enough overlap)



# Cat model

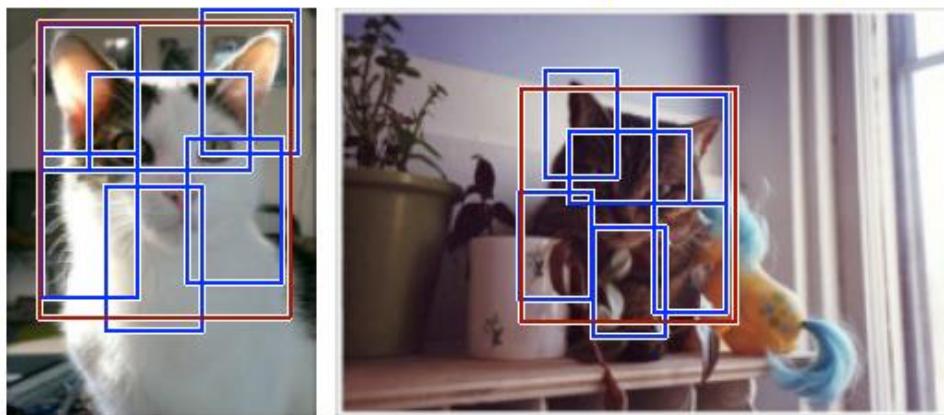
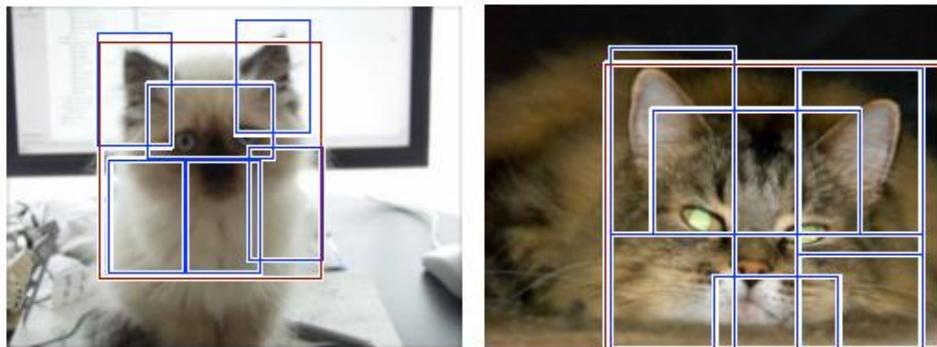
---



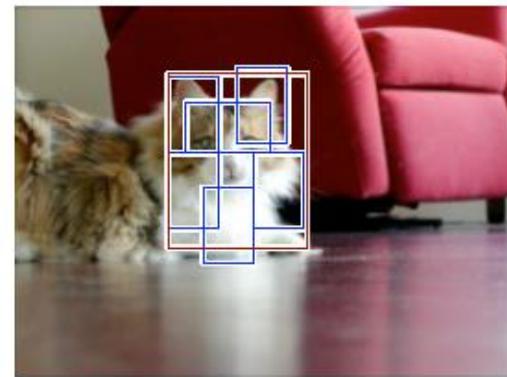
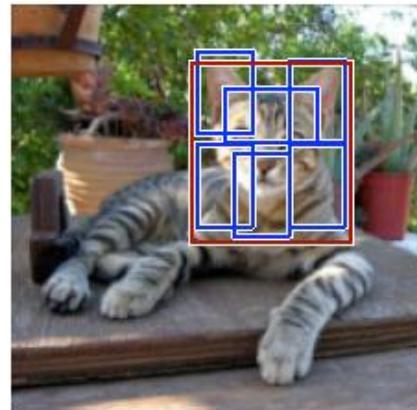
# Cat detections

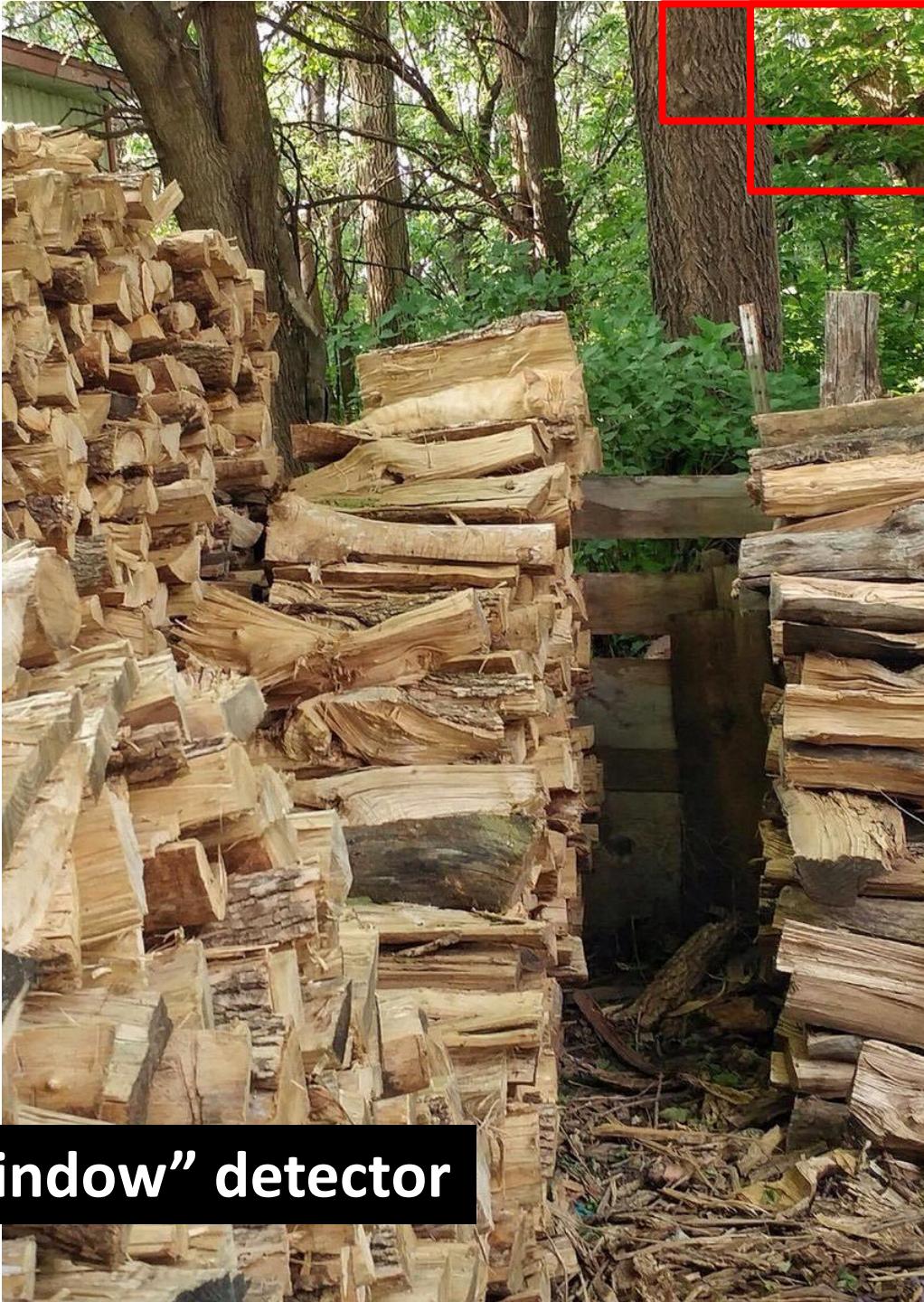
---

high scoring true positives



high scoring false positives  
(not enough overlap)





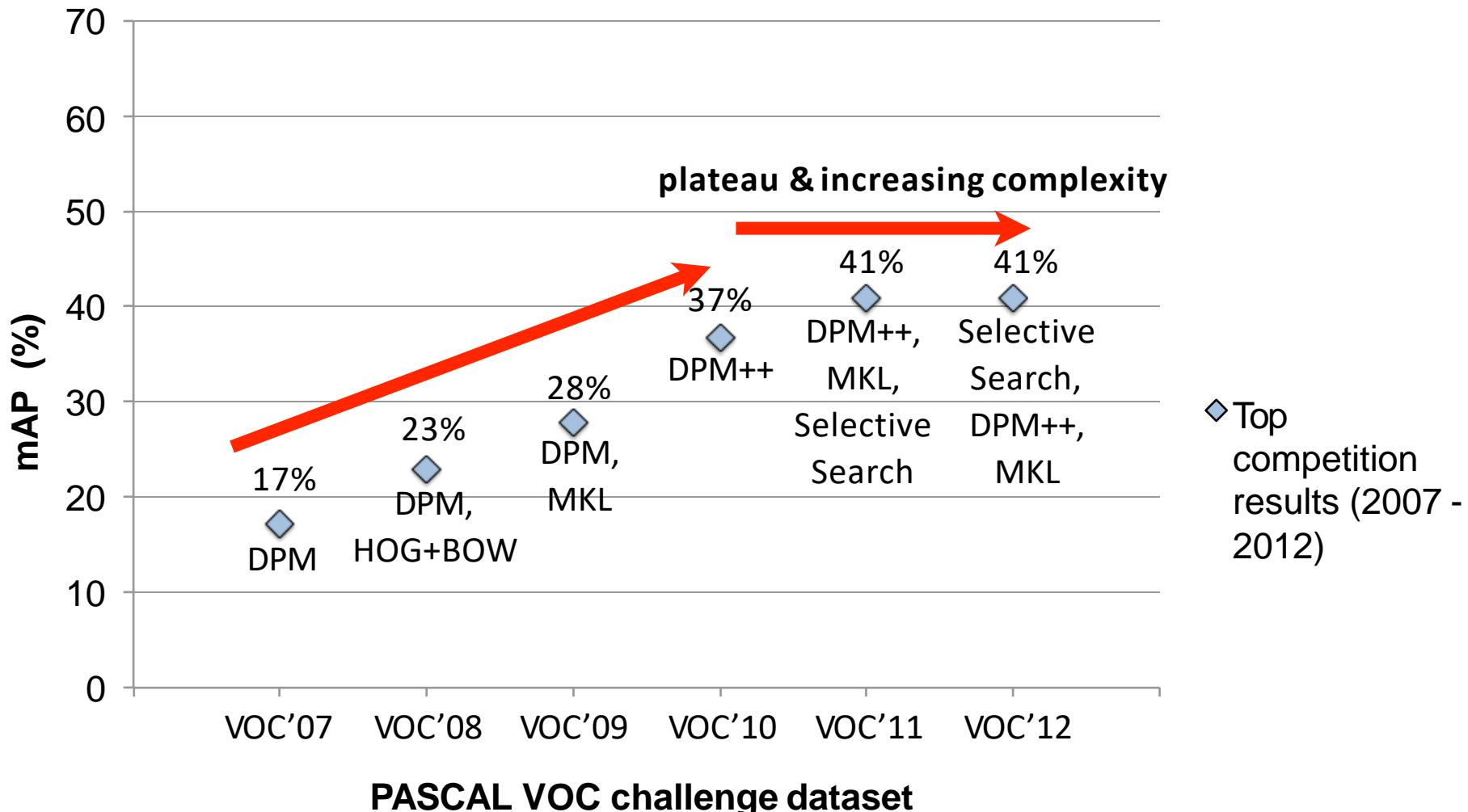
**“Sliding window” detector**

# Plan for the next three lectures

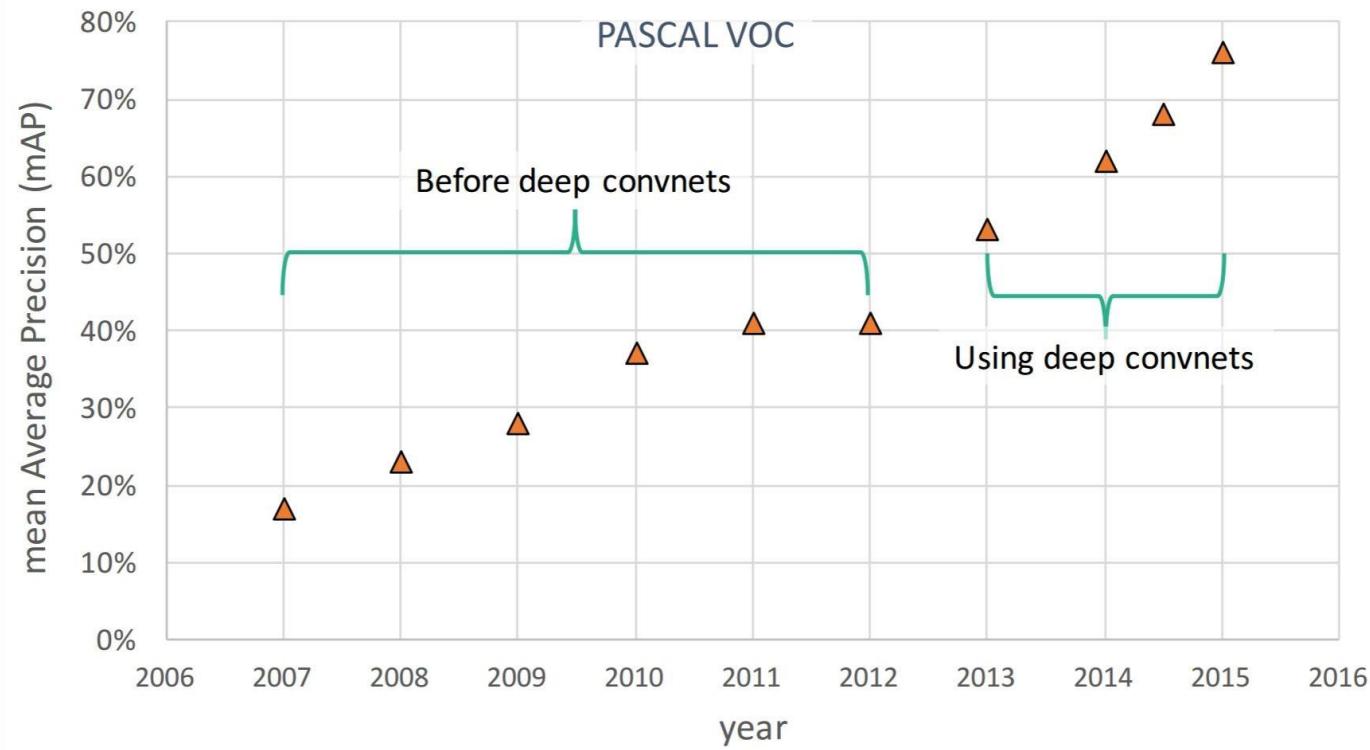
- Detection approaches
  - Pre-CNNs
    - Detection with whole windows: Pedestrian detection
    - Part-based detection: Deformable Part Models
  - Post-CNNs
    - Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
    - Detection without region proposals: YOLO, SSD
- Segmentation approaches
  - Semantic segmentation: FCN
  - Instance segmentation: Mask R-CNN

# Complexity and the plateau

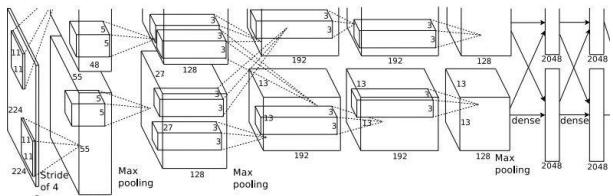
[Source: <http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc20{07,08,09,10,11,12}/results/index.html>]



# Impact of Deep Learning



# Before: Image Classification with CNNs

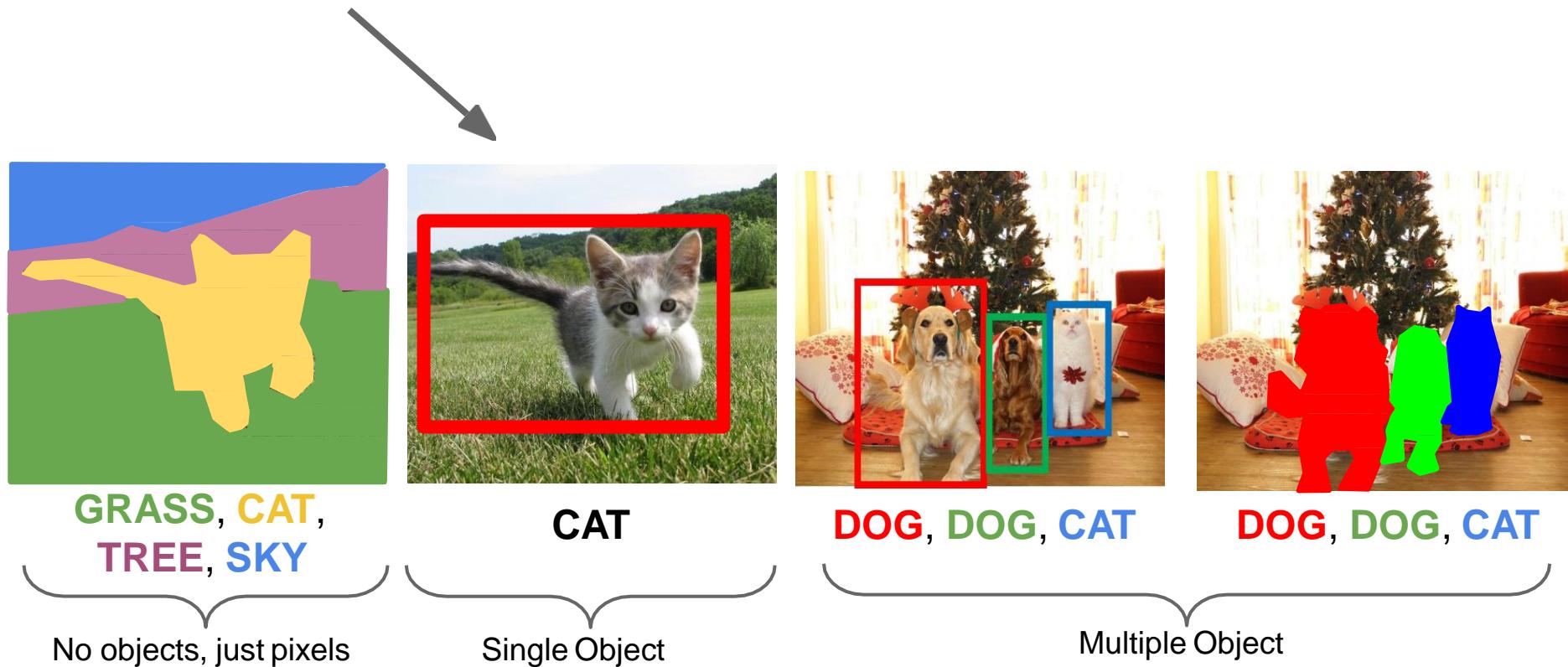


**Vector:**  
4096

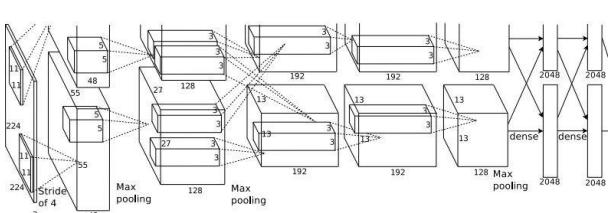
**Fully-Connected:**  
4096 to 1000

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

# Classification + Localization



# Classification + Localization



Fully  
Connected:  
4096 to 1000

## Class Scores

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

Vector: Fully  
Connected:  
4096 to 4

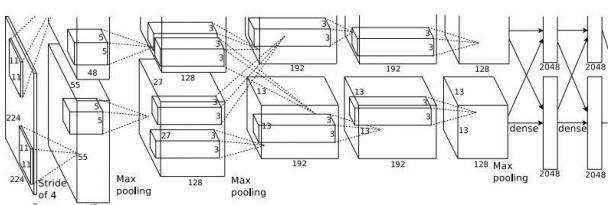
## Box Coordinates

(x, y, w, h)

Treat localization as a  
regression problem!



# Classification + Localization



Treat localization as a regression problem!

Fully  
Connected:  
4096 to 1000

Vector: Fully  
Connected:  
4096 to 4

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

Correct label:  
Cat

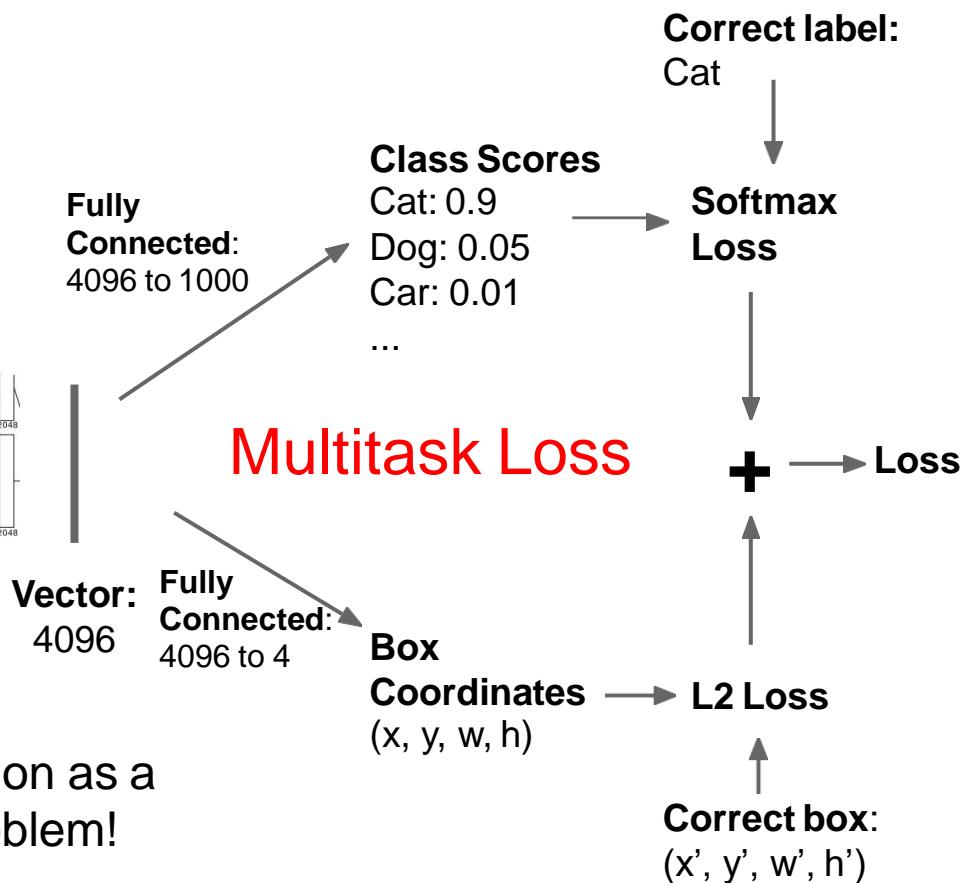
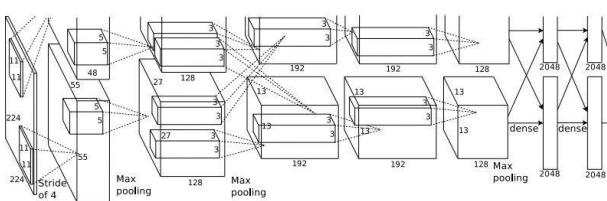
Softmax  
Loss



Box  
Coordinates  $\rightarrow$  L2 Loss  
( $x, y, w, h$ )

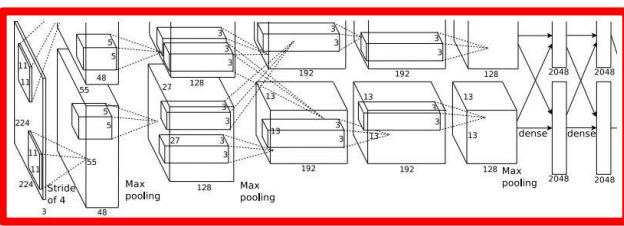
Correct box:  
( $x', y', w', h'$ )

# Classification + Localization



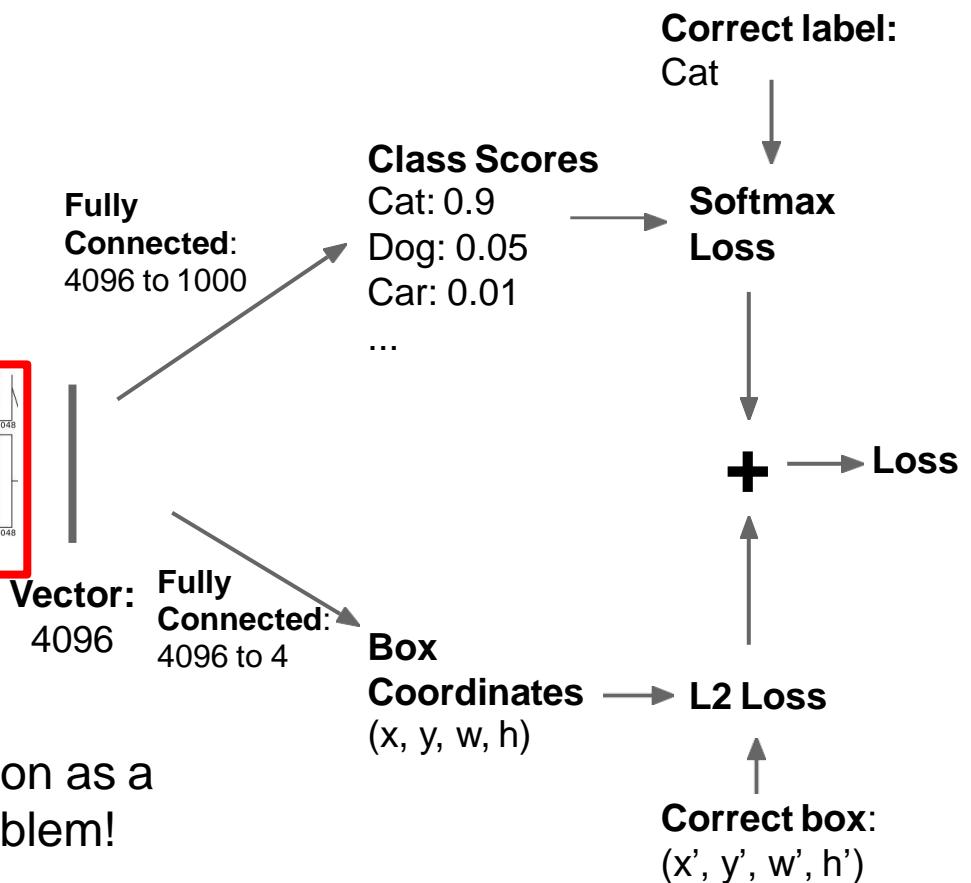
Treat localization as a regression problem!

# Classification + Localization

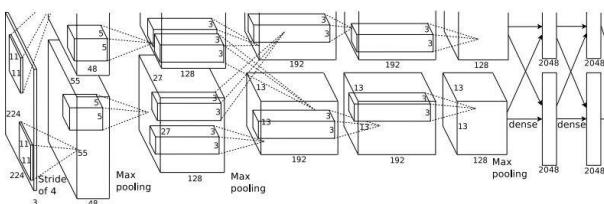
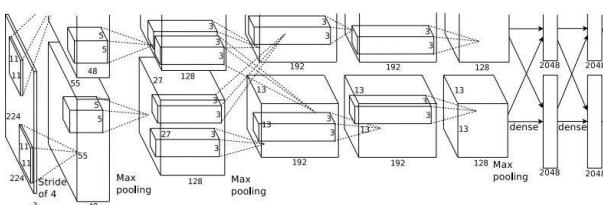
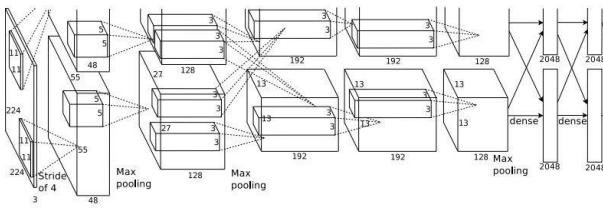


Often pretrained on ImageNet  
(Transfer learning)

Treat localization as a  
regression problem!



# Object Detection as Regression?



CAT: (x, y, w, h)

DOG: (x, y, w, h)

DOG: (x, y, w, h)

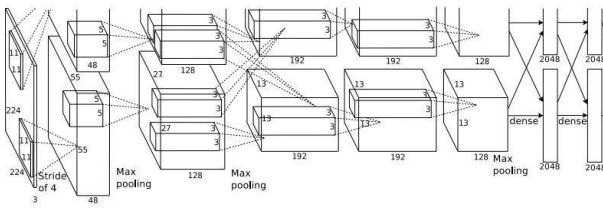
CAT: (x, y, w, h)

DUCK: (x, y, w, h)

DUCK: (x, y, w, h)

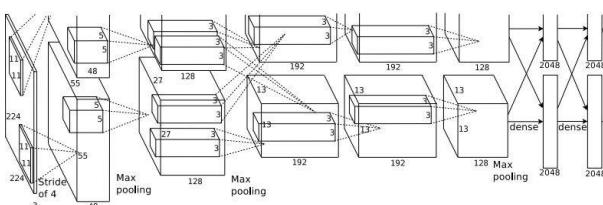
...

# Object Detection as Regression?



CAT: (x, y, w, h)

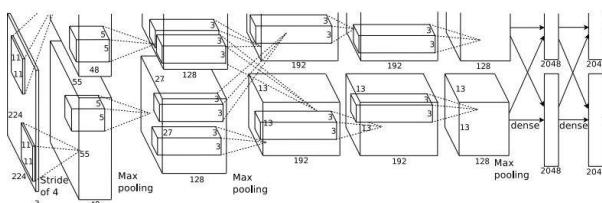
4 numbers



DOG: (x, y, w, h)

16 numbers

CAT: (x, y, w, h)



DUCK: (x, y, w, h)

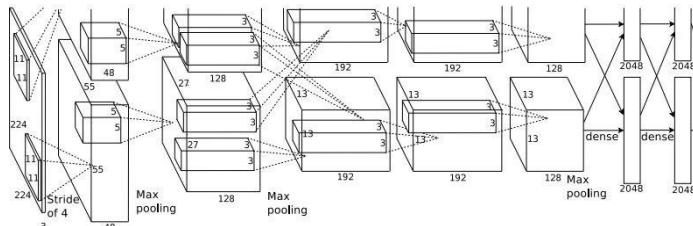
Many  
numbers!

....

Each image needs a different  
number of outputs!

# Object Detection as Classification: Sliding Window

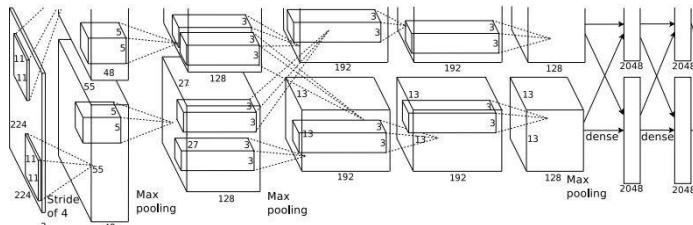
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO  
Cat? NO  
Background? YES

# Object Detection as Classification: Sliding Window

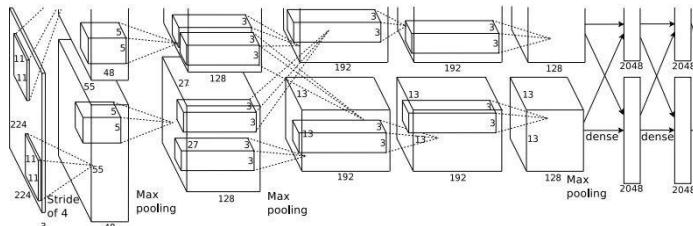
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES  
Cat? NO  
Background? NO

# Object Detection as Classification: Sliding Window

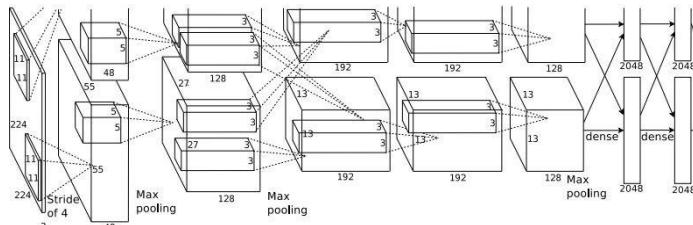
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES  
Cat? NO  
Background? NO

# Object Detection as Classification: Sliding Window

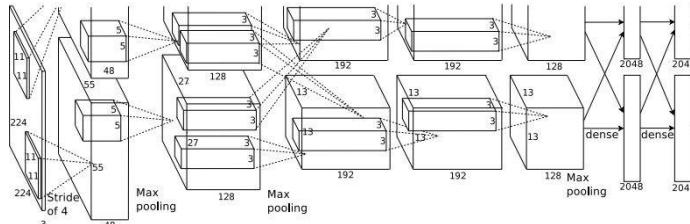
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO  
Cat? YES  
Background? NO

# Object Detection as Classification: Sliding Window

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

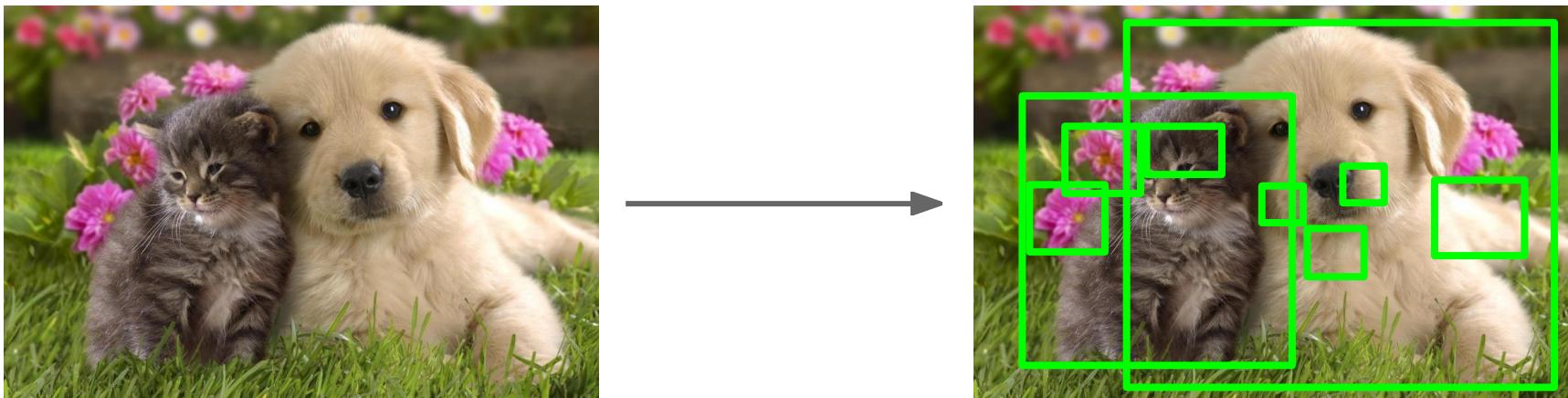


Dog? NO  
Cat? YES  
Background? NO

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

# 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



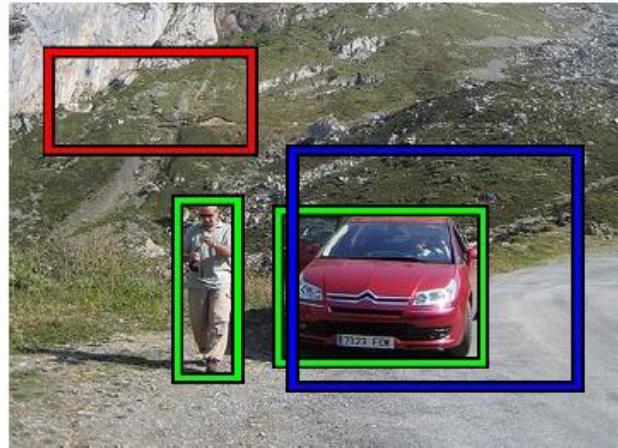
Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012

Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

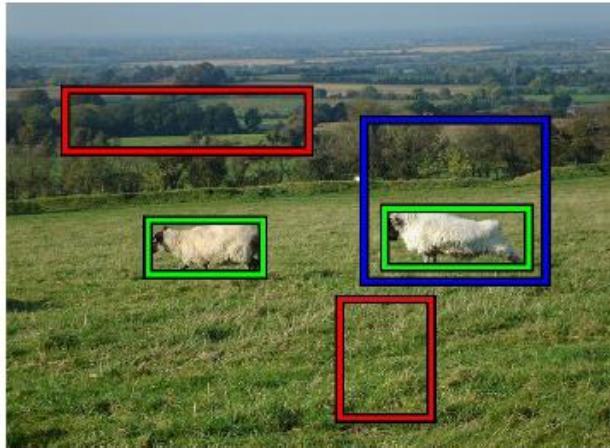
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014

Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

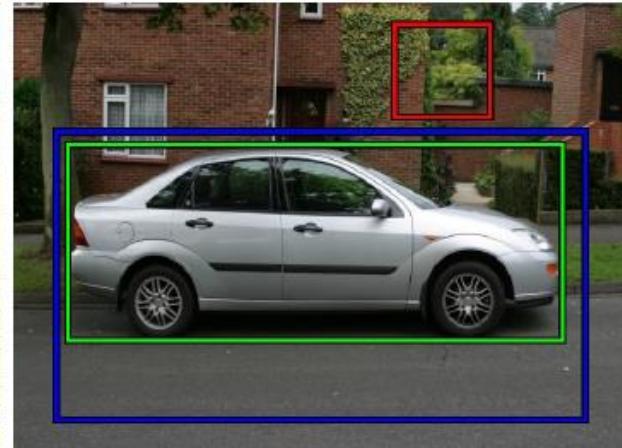
# Speeding up detection: Restrict set of windows we pass through SVM to those w/ high “objectness”



(a)



(b)



(c)

Fig. 1: **Desired behavior of an objectness measure.** *The desired objectness measure should score the blue windows, partially covering the objects, lower than the ground truth windows (green), and score even lower the red windows containing only stuff or small parts of objects.*

# Objectness cue #1: Where people look



Fig. 2: MS success and failure.

# Objectness cue #2: color contrast at boundary

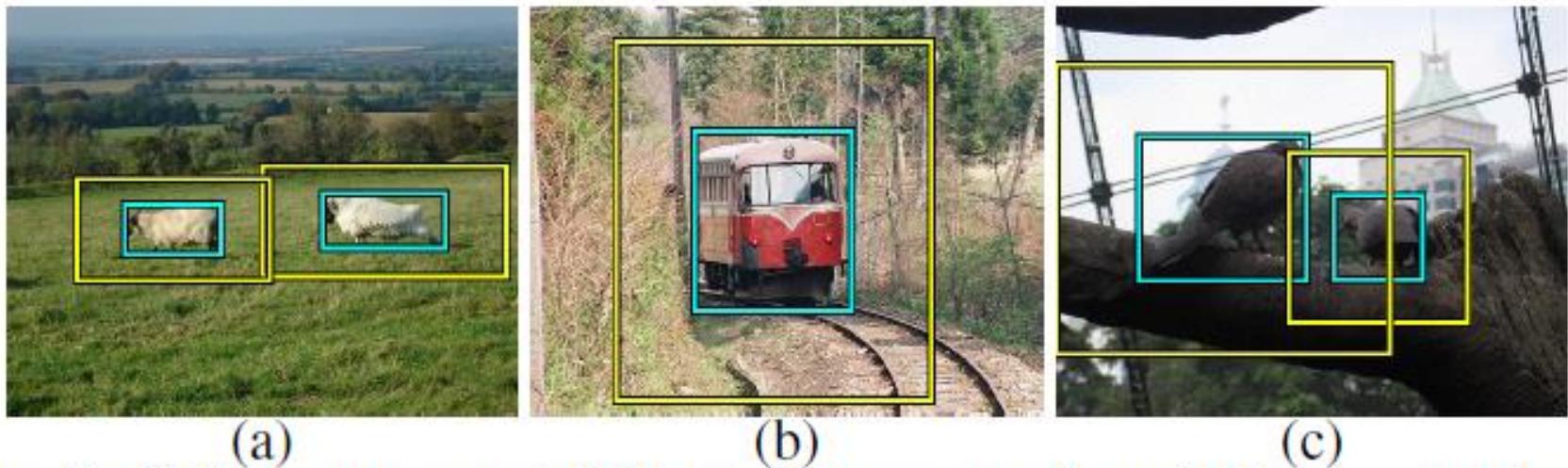


Fig. 3: **CC success and failure.** **Success:** the windows containing the objects (cyan) have high color contrast with their surrounding ring (yellow) in images (a) and (b). **Failure:** the color contrast for windows in cyan in image (c) is much lower.

# Objectness cue #3: no segments “straddling” the object box

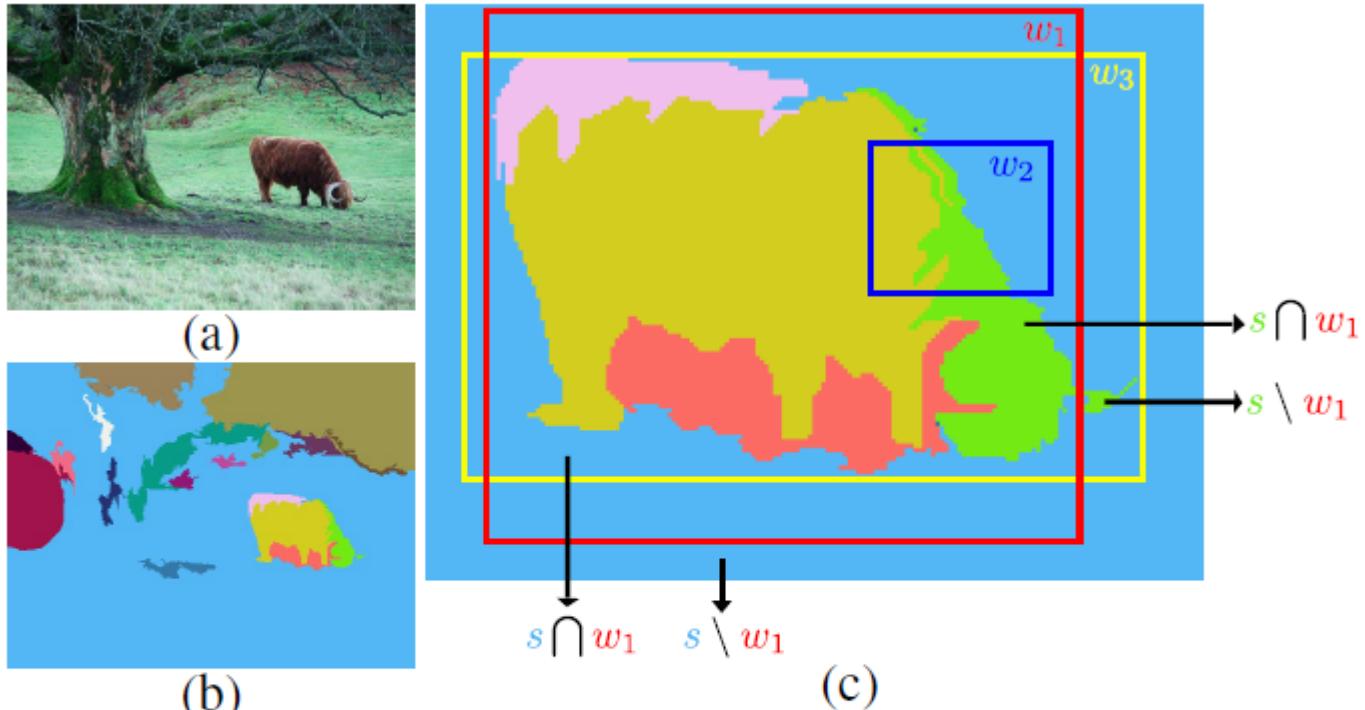
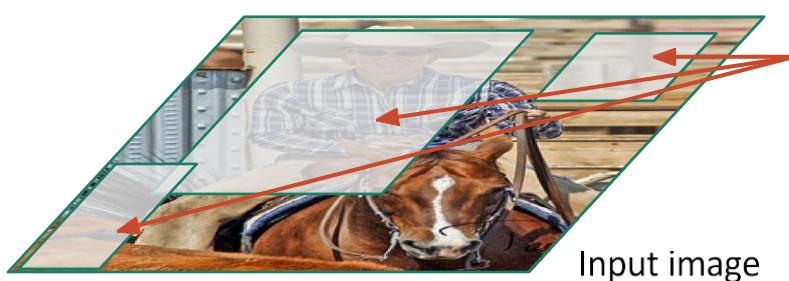


Fig. 5: **The SS cue.** Given the segmentation (b) of image (a), for a window  $w$  we compute  $\text{SS}(w, \theta_{\text{SS}})$  (eq. 4). In (c), most of the surface of  $w_1$  is covered by superpixels contained almost entirely inside it. Instead, all superpixels passing by  $w_2$  continue largely outside it. Therefore,  $w_1$  has a higher SS score than  $w_2$ . The window  $w_3$  has an even higher score as it fits the object tightly.

# R-CNN

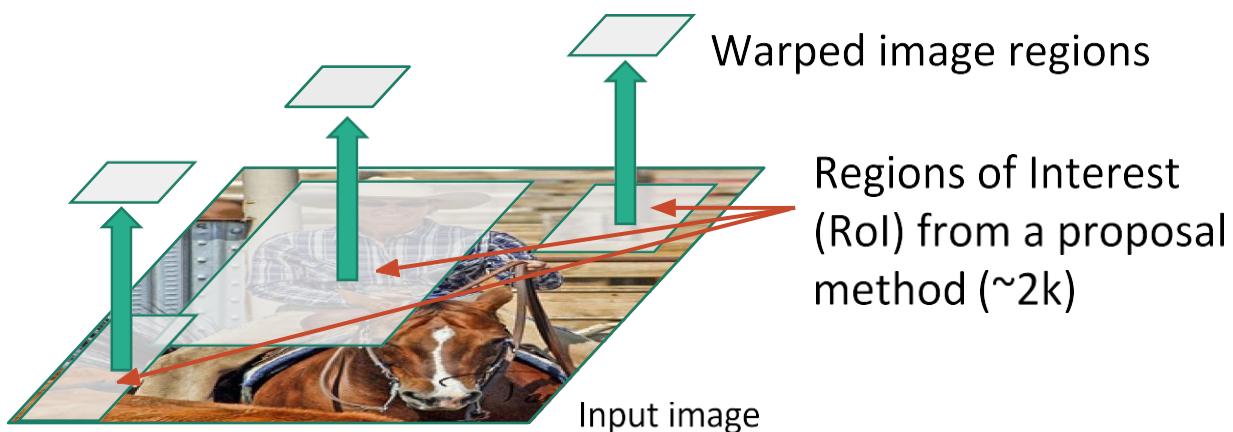


# R-CNN

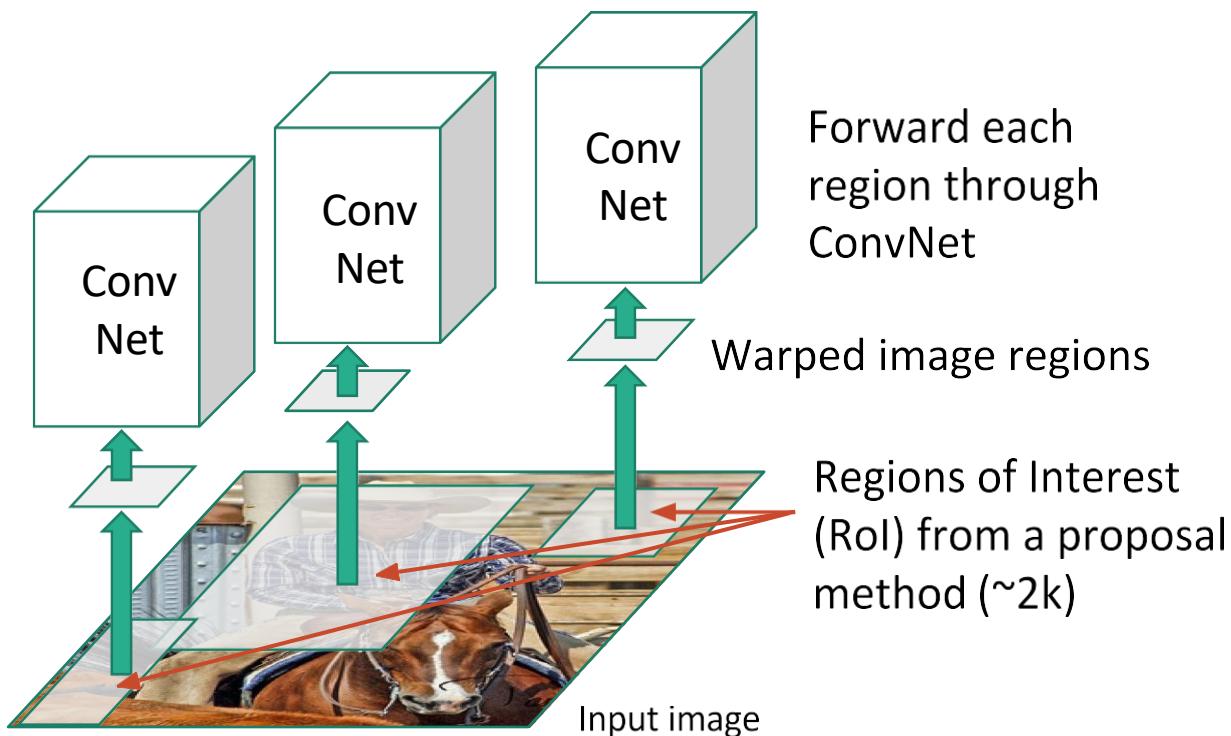


Regions of Interest  
(RoI) from a proposal  
method (~2k)

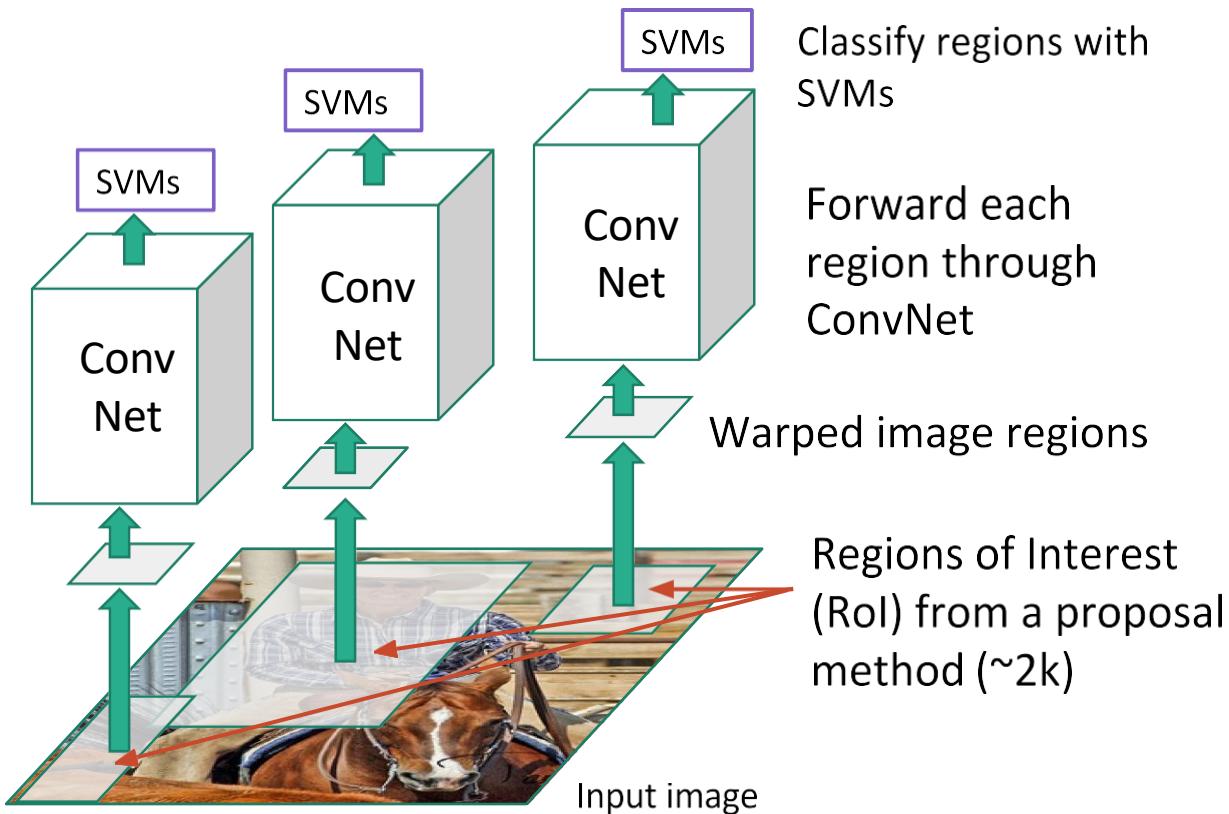
# R-CNN



# R-CNN

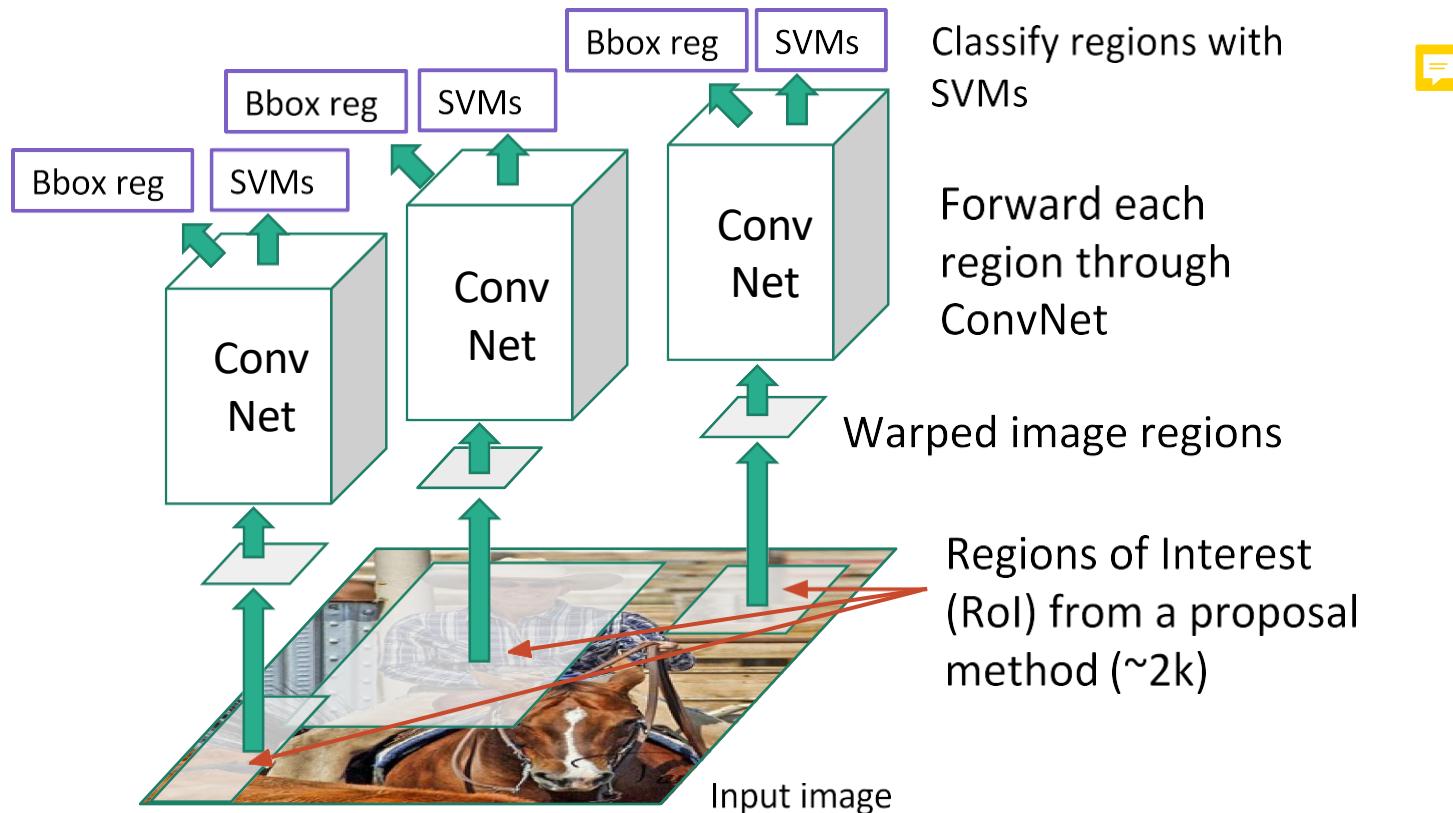


# R-CNN

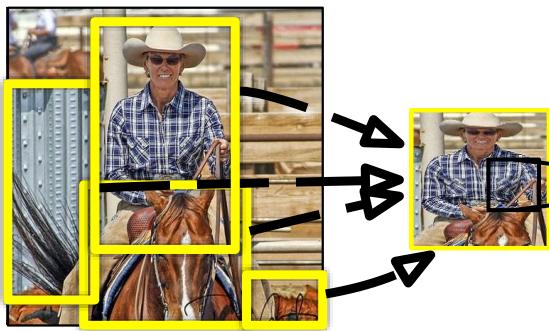
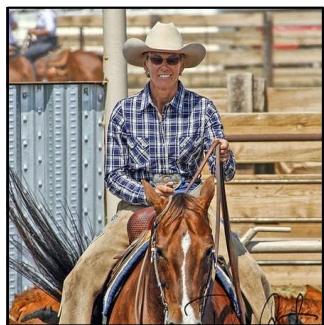


# R-CNN

Linear Regression for bounding box offsets

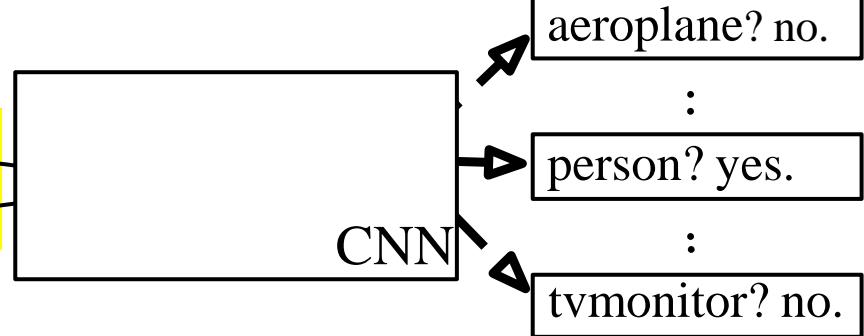


# R-CNN: Regions with CNN features



Input  
image

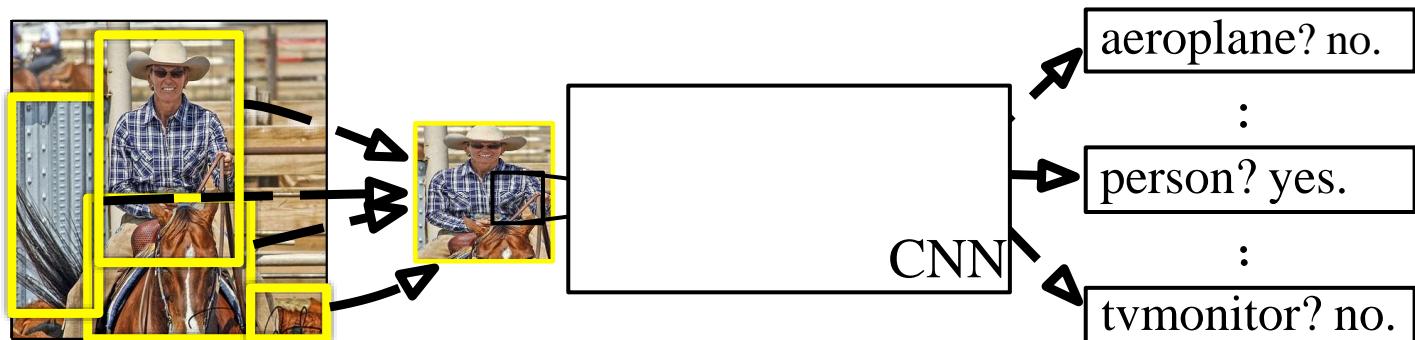
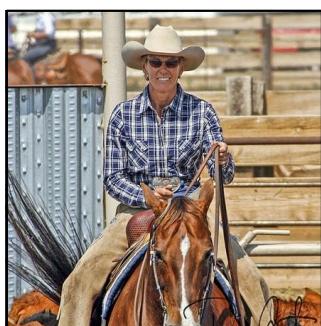
Extract region  
proposals (~2k / image)



Compute CNN  
features

Classify regions  
(linear SVM)

# R-CNN at test time: Step 1

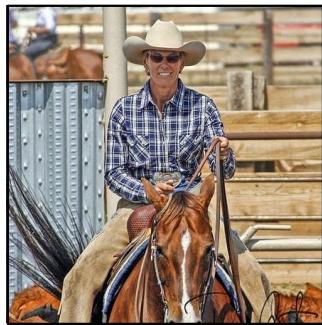


Input  
image → Extract region  
proposals (~2k / image)

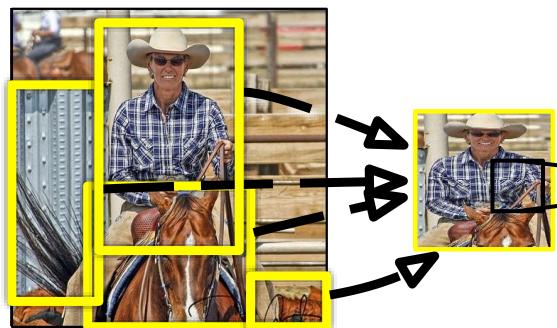
Proposal-method agnostic, many choices

- Selective Search [van de Sande, Uijlings et al.] (Used in this work)
- Objectness [Alexe et al.]
- Category independent object proposals [Endres & Hoiem]
- CPMC [Carreira & Sminchisescu]

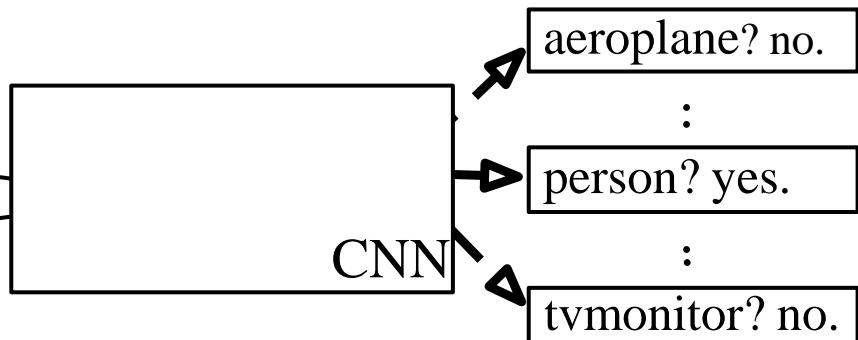
# R-CNN at test time: Step 2



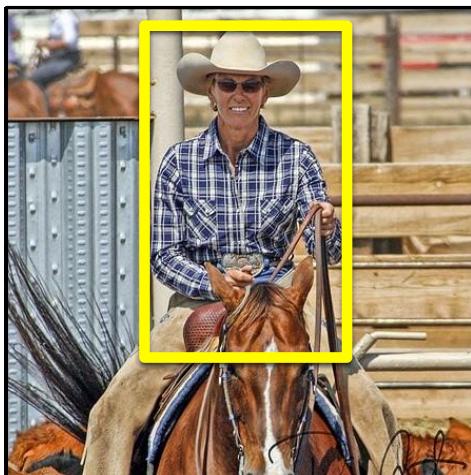
Input  
image



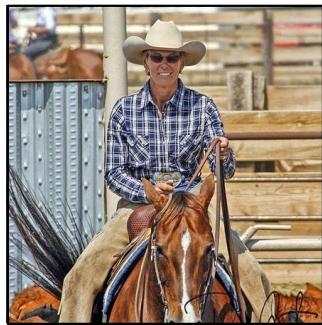
Extract region  
proposals (~2k / image)



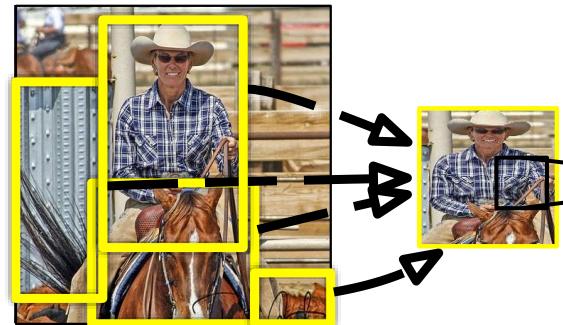
Compute CNN  
features



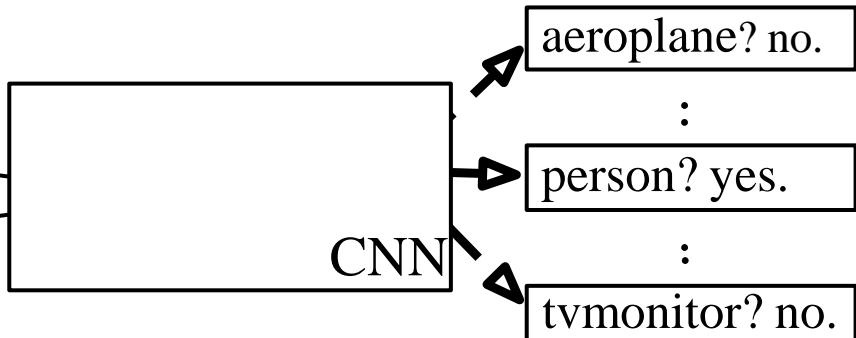
# R-CNN at test time: Step 2



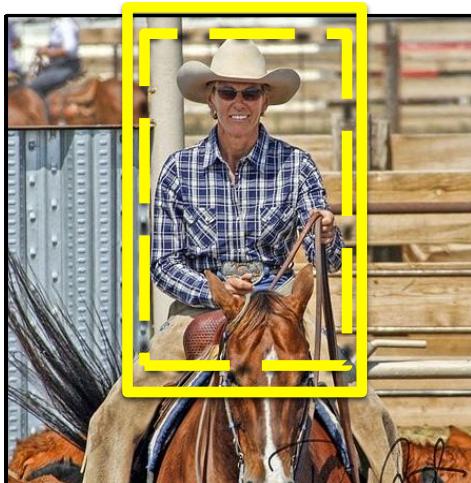
Input  
image



Extract region  
proposals (~2k / image)

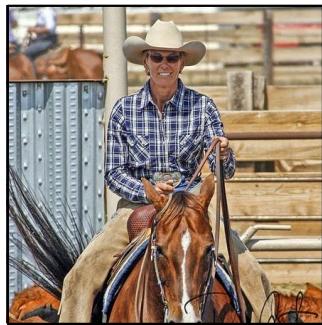


Compute CNN  
features

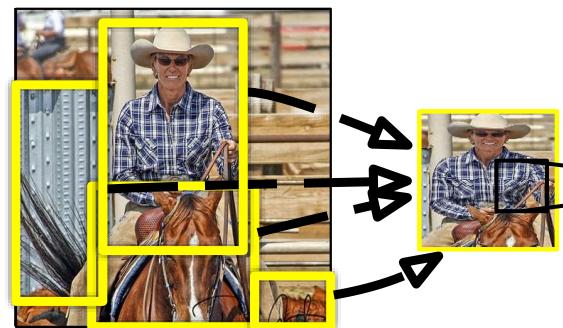


Dilate proposal

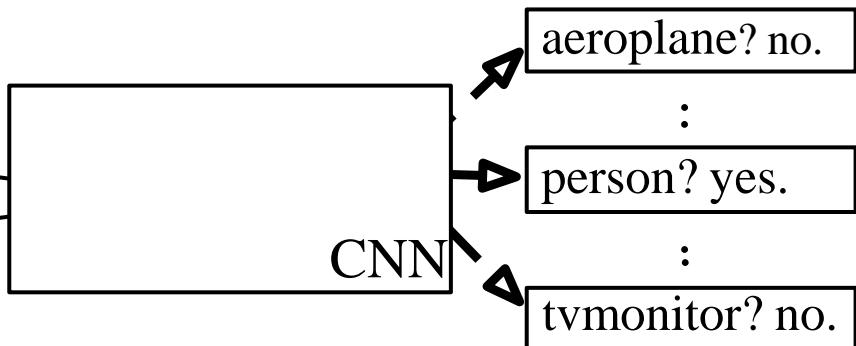
# R-CNN at test time: Step 2



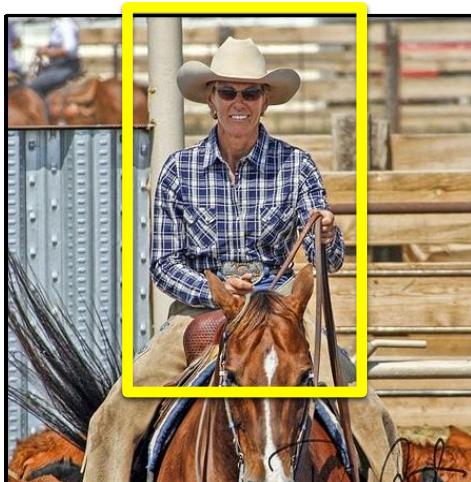
Input  
image



Extract region  
proposals (~2k / image)

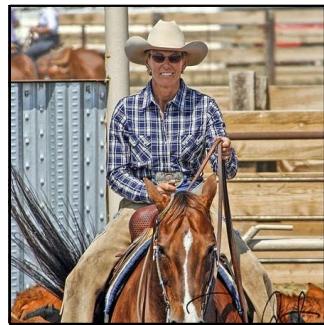


Compute CNN  
features

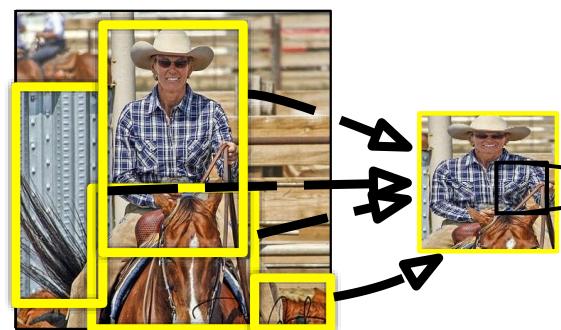


a. Crop

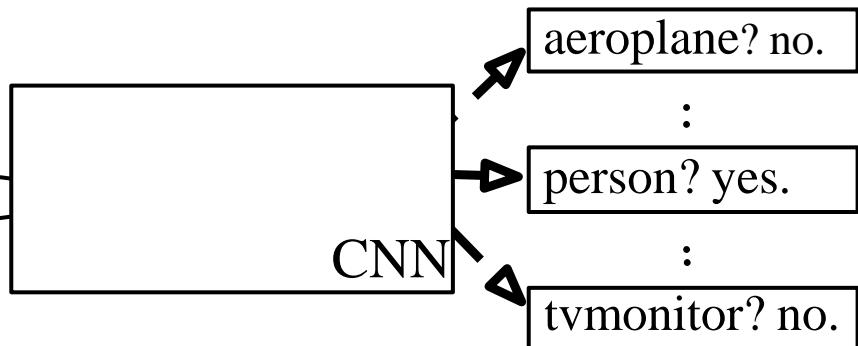
# R-CNN at test time: Step 2



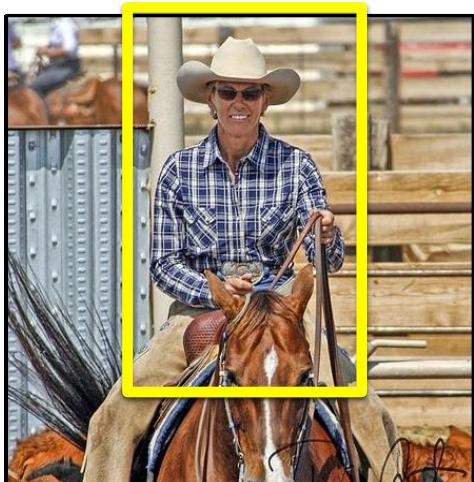
Input  
image



Extract region  
proposals (~2k / image)



Compute CNN  
features



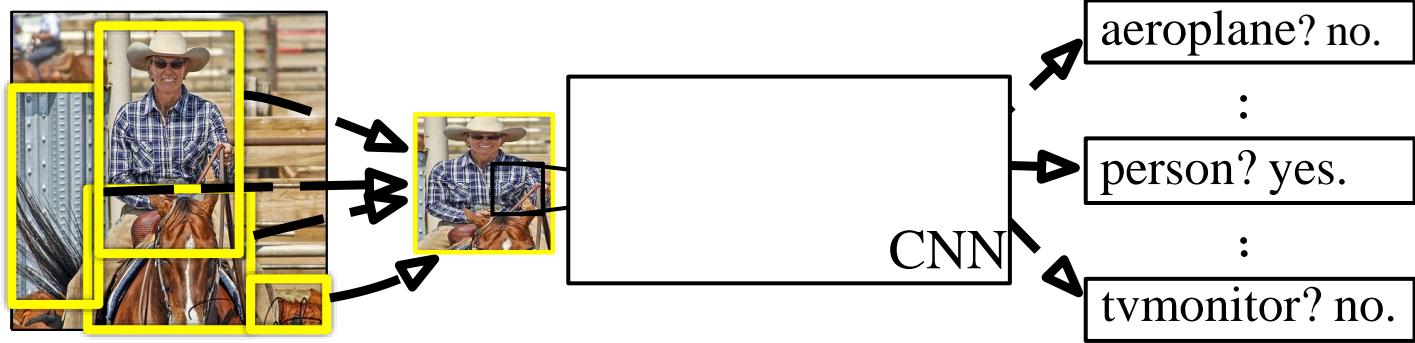
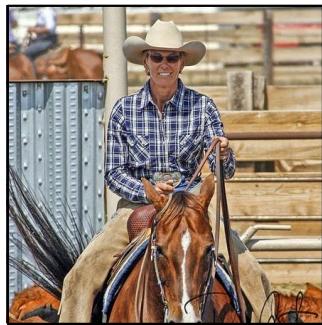
a. Crop



b. Scale (anisotropic)

227 x 227

# R-CNN at test time: Step 2



Input  
image

Extract region  
proposals (~2k / image)

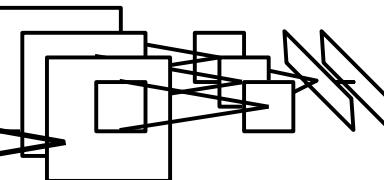
Compute CNN  
features



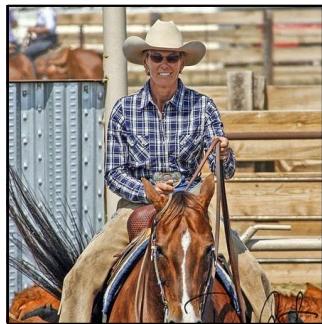
c. Crop

b. Scale (anisotropic)

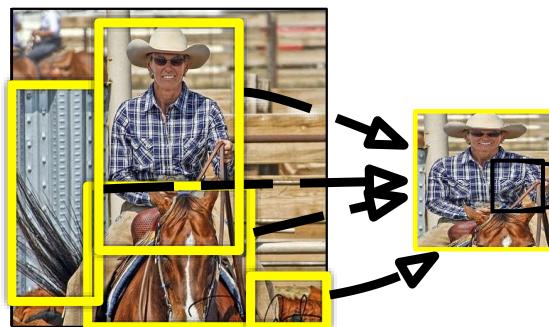
c. Forward propagate  
Output: "fc<sub>7</sub>" features



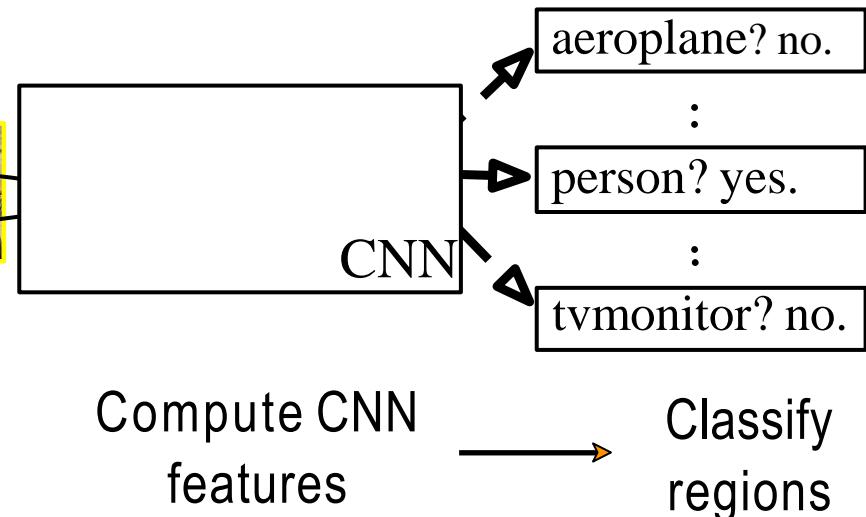
# R-CNN at test time: Step 3



Input  
image



Extract region  
proposals (~2k / image)



Compute CNN  
features

Classify  
regions

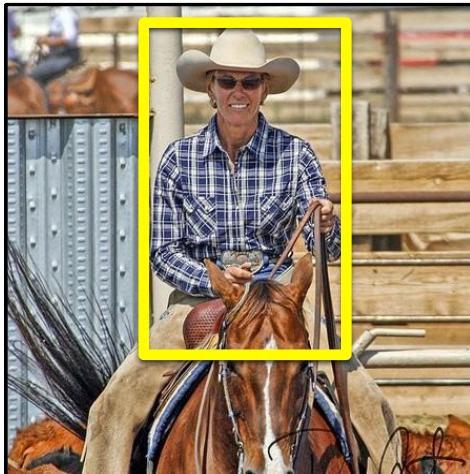


proposal

4096-dimensional  
fc<sub>7</sub> feature vector

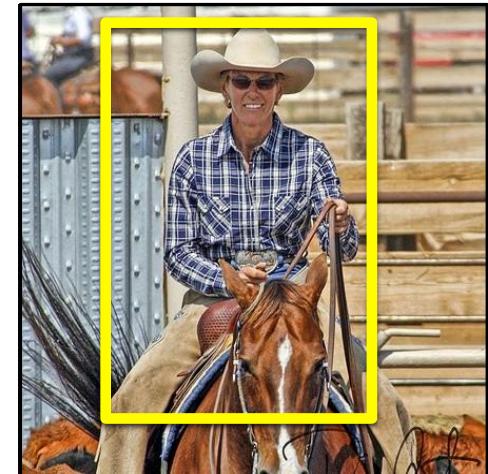
linear classifiers  
(SVM or softmax)

# Step 4: Object proposal refinement



Original  
proposal

Linear regression  
on CNN features

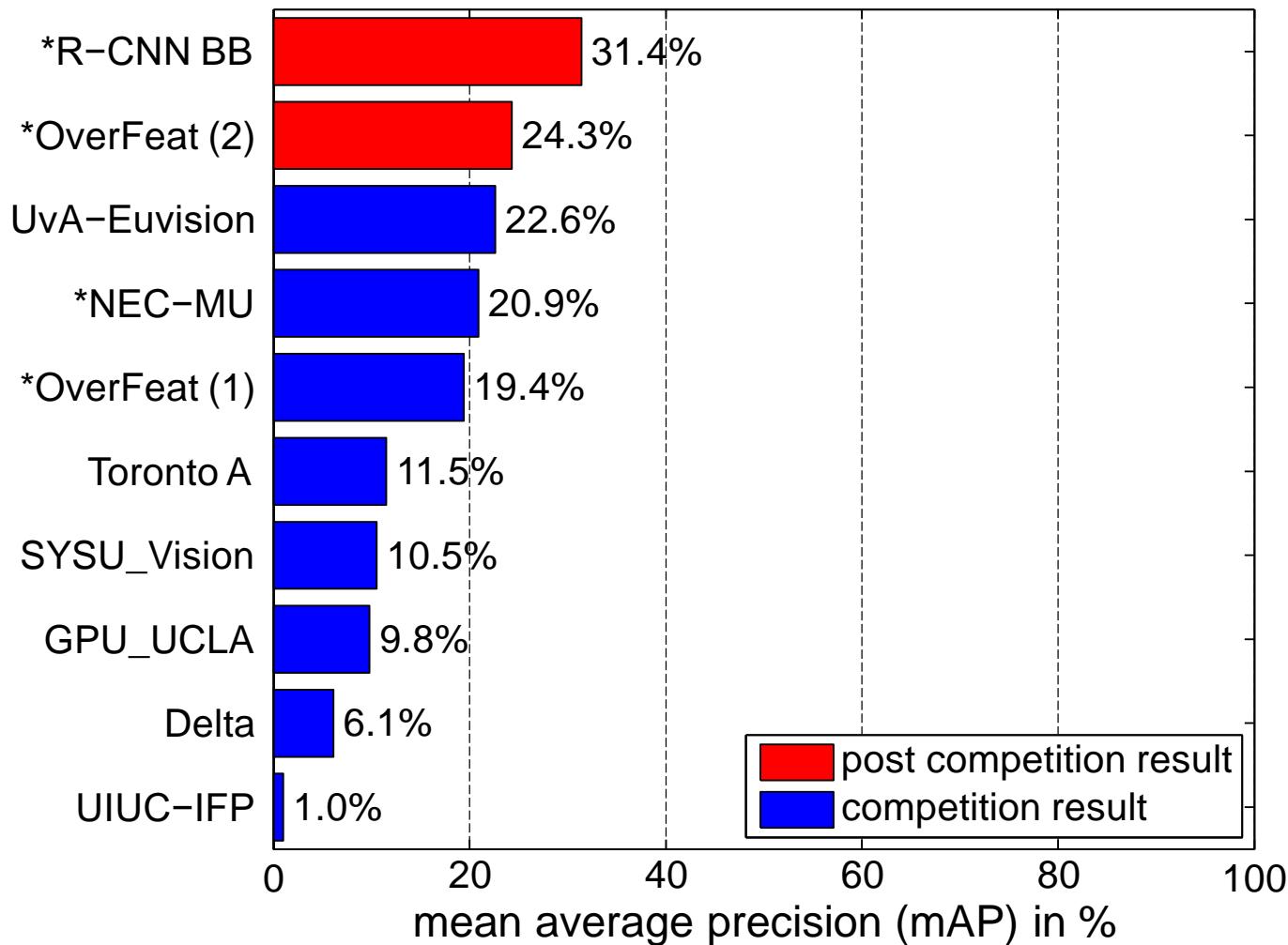


Predicted  
object bounding box

Bounding-box regression

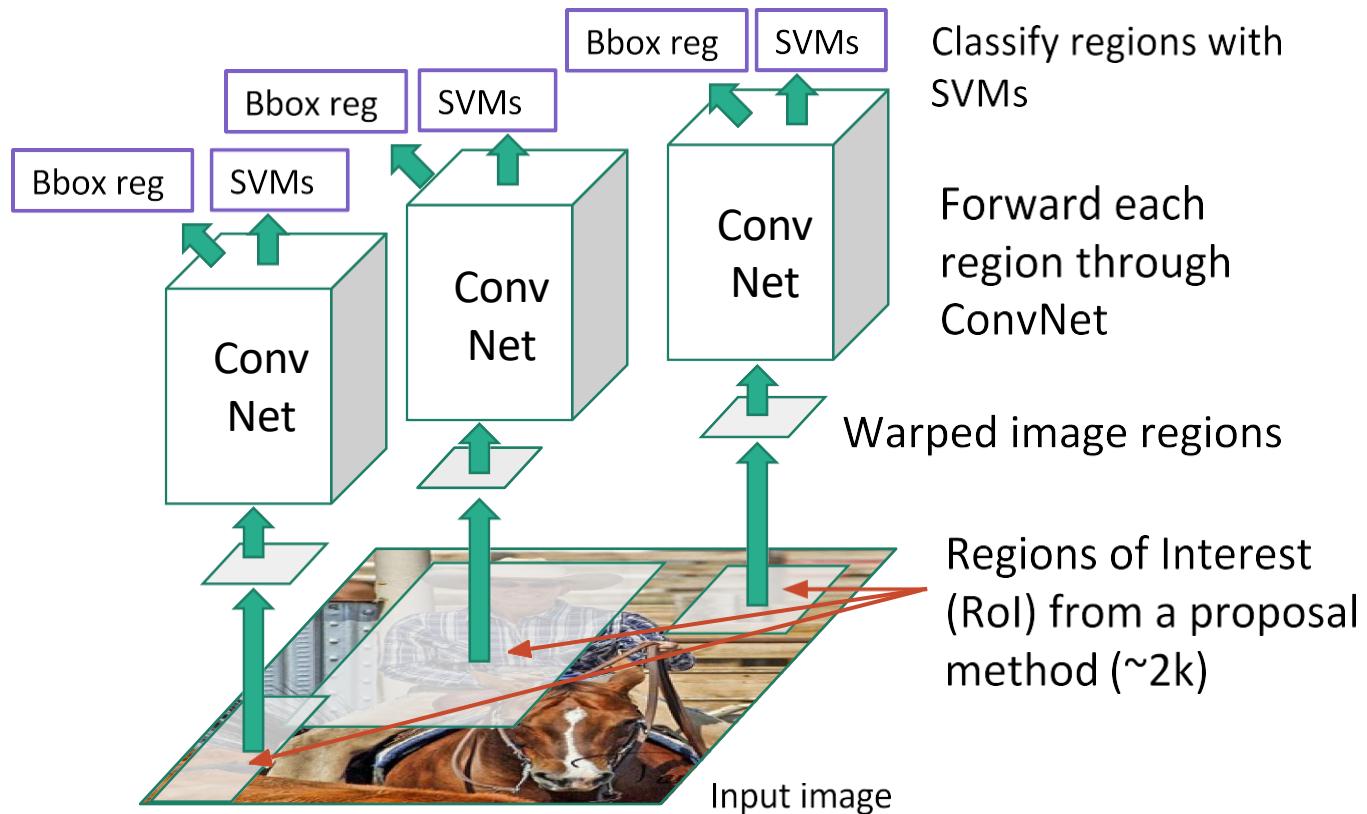
# R-CNN on ImageNet detection

ILSVRC2013 detection test set mAP



# R-CNN

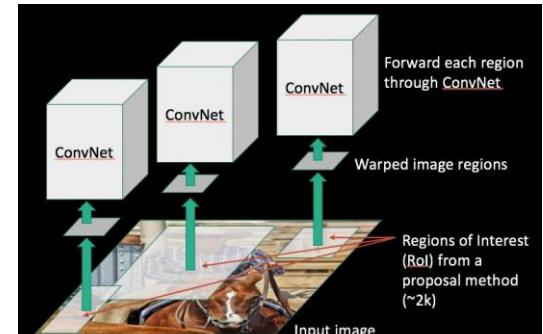
Linear Regression for bounding box offsets



Post hoc component

# What's wrong with slow R-CNN?

- Ad-hoc training objectives
  - Train post-hoc linear SVMs (hinge loss)
  - Train post-hoc bounding-box regressions (L2 loss)
- Training is slow (84h), takes a lot of disk space
  - Need to store all region crops
- Inference (detection) is slow
  - 47s / image with VGG16 [Simonyan & Zisserman, ICLR15]



~2000 ConvNet forward passes per image

# Fast R-CNN

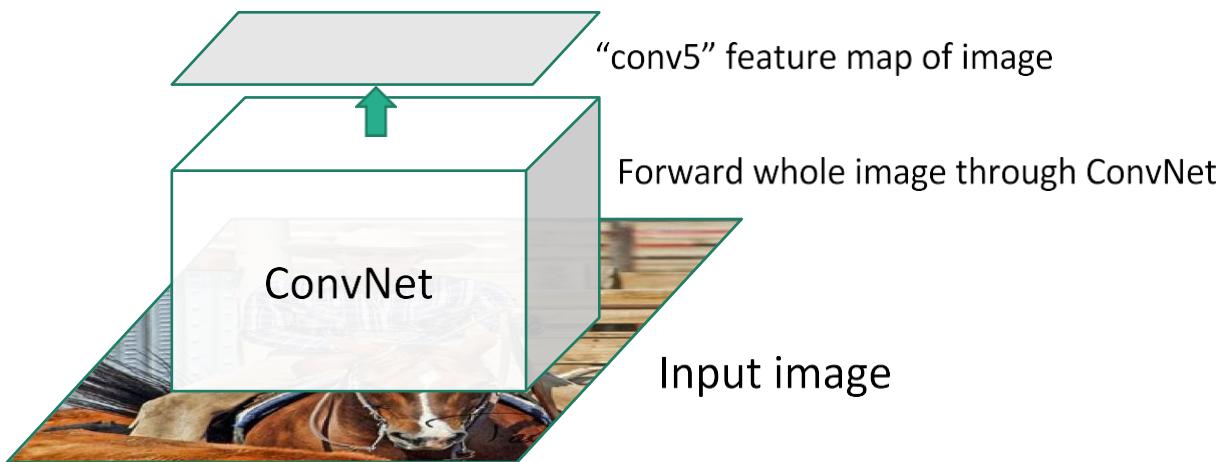
- One network, applied one time, not 2000 times
- Trained end-to-end (in one stage)
- Fast test time
- Higher mean average precision

# Fast R-CNN

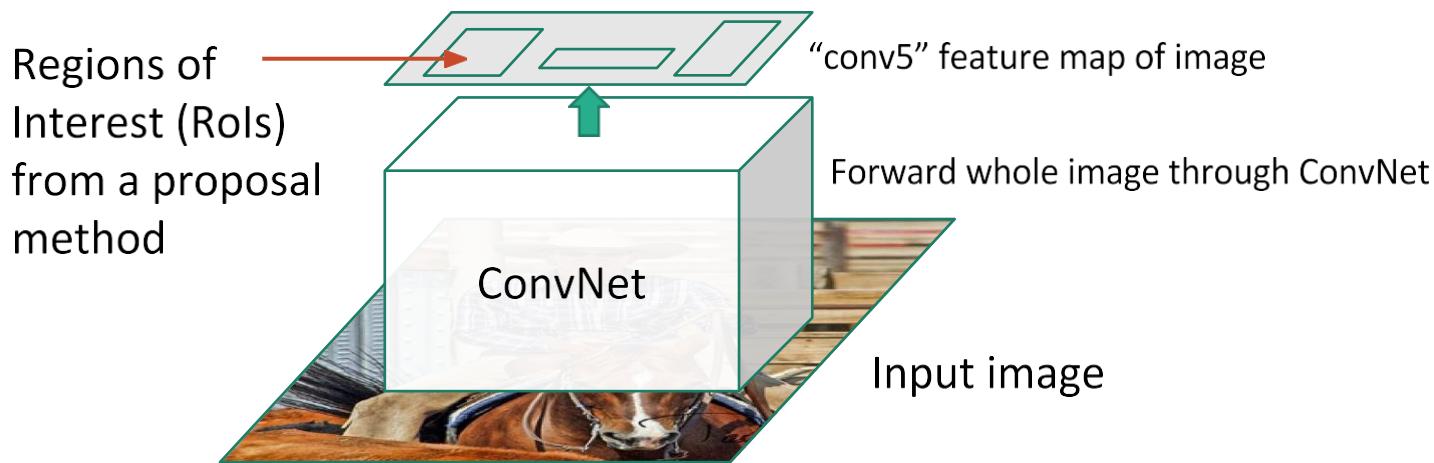


Input image

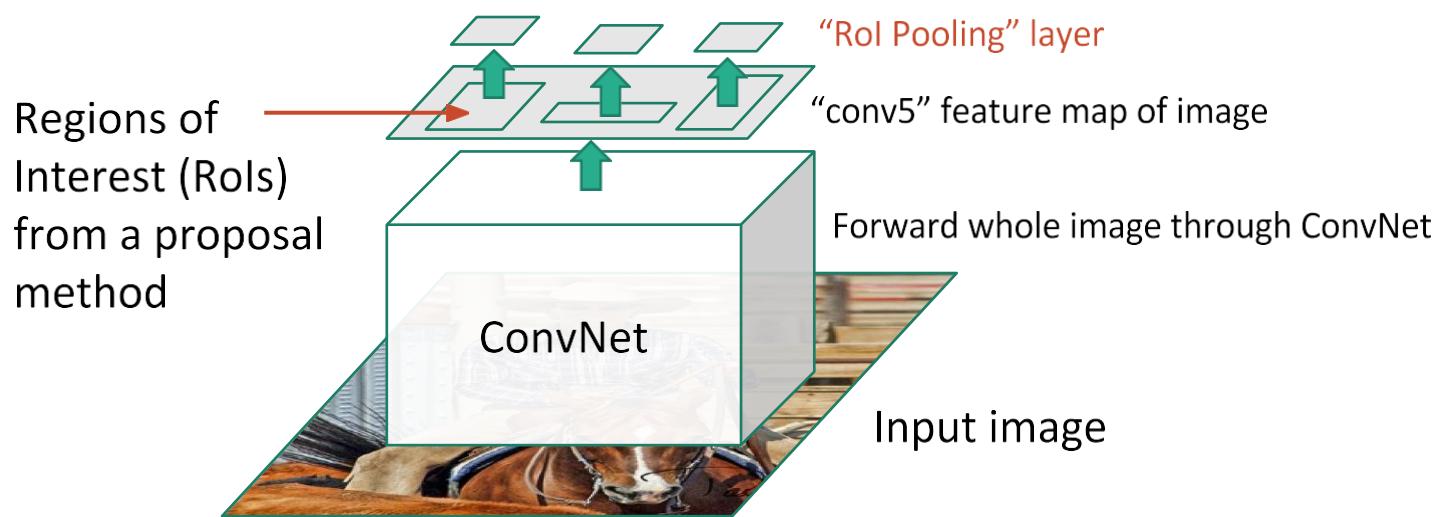
# Fast R-CNN



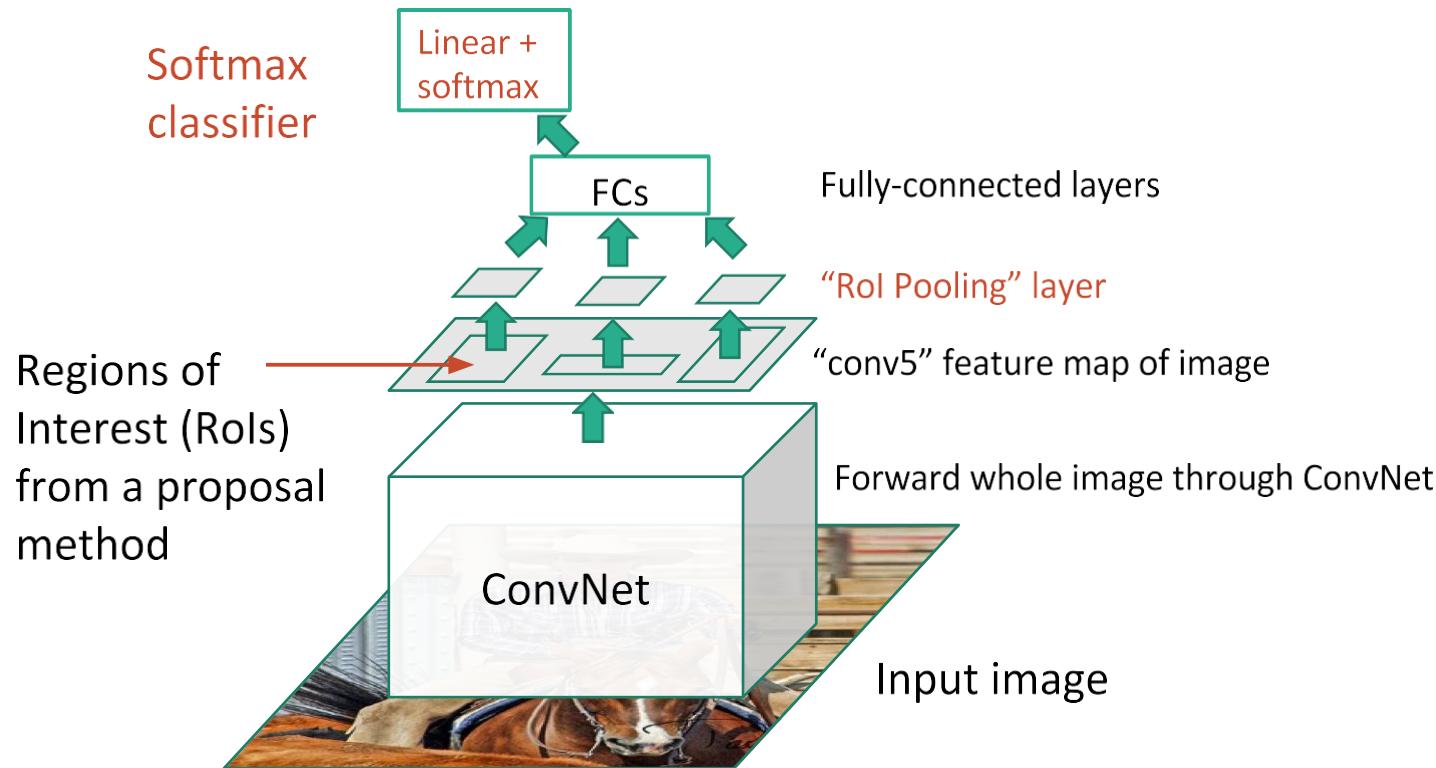
# Fast R-CNN



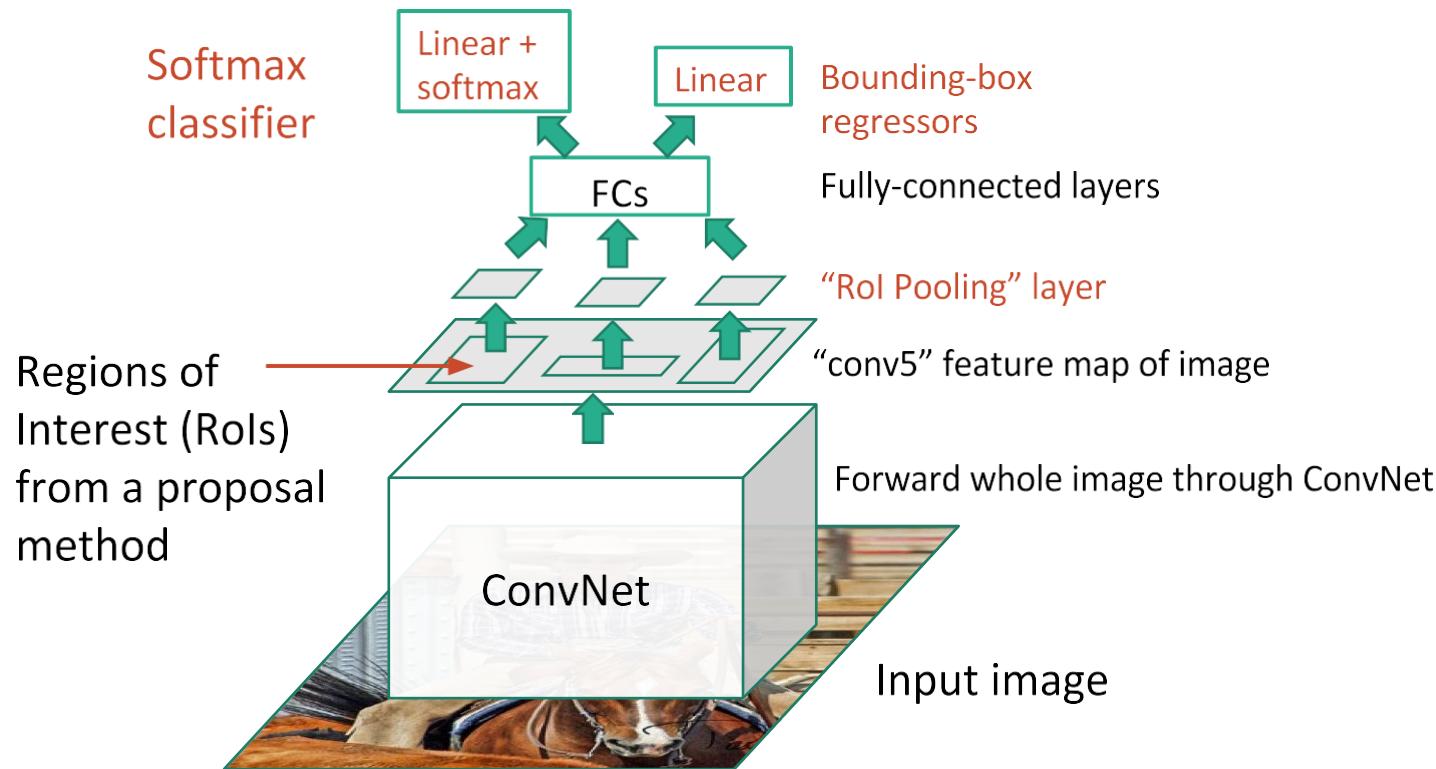
# Fast R-CNN



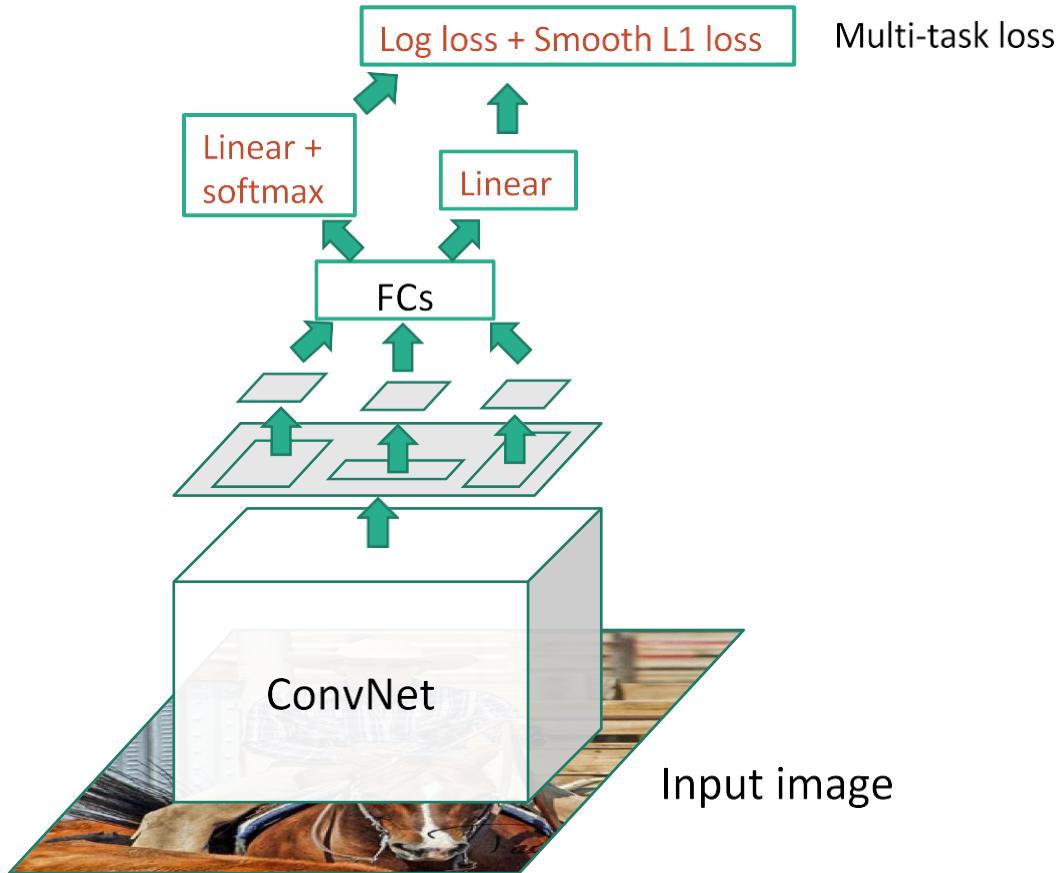
# Fast R-CNN



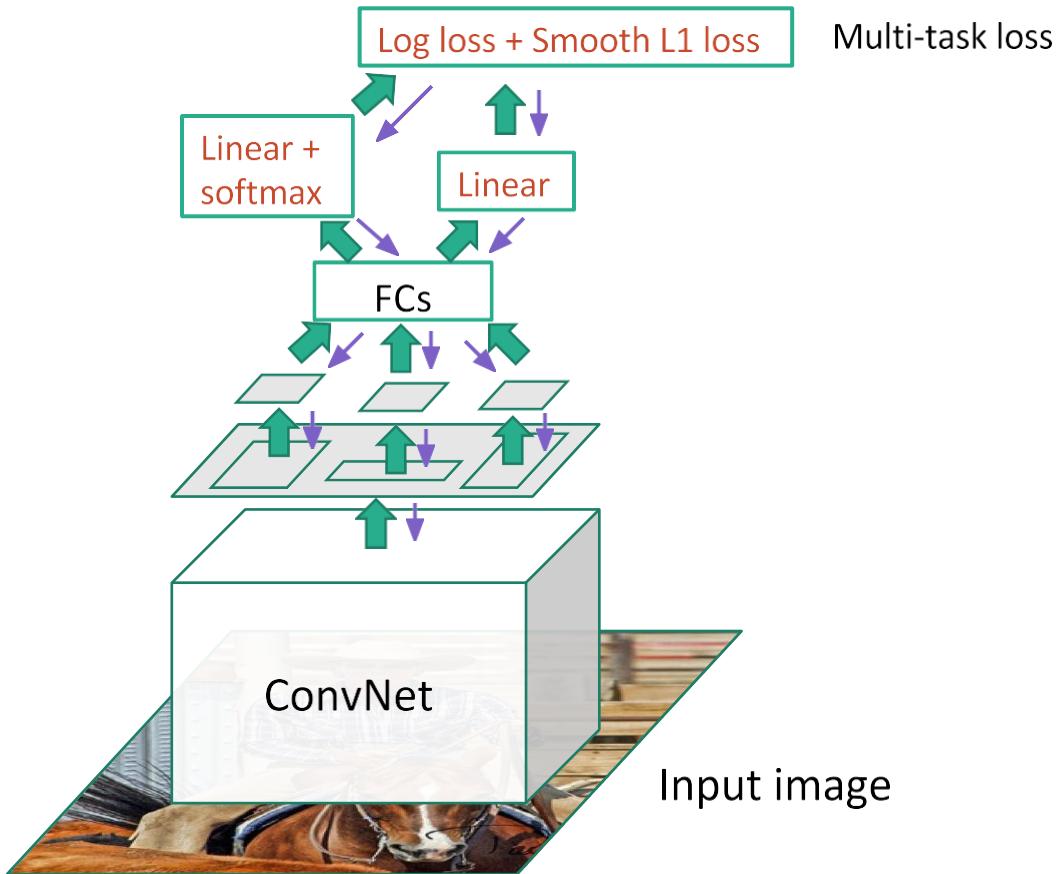
# Fast R-CNN



# Fast R-CNN (Training)



# Fast R-CNN (Training)



# Fast R-CNN vs R-CNN

	Fast R-CNN	R-CNN
Train time (h)	9.5	84
Speedup	8.8x	1x
Test time / image	0.32s	47.0s
Test speedup	146x	1x
mAP	66.9%	66.0%

Timings exclude object proposal time, which is equal for all methods.  
All methods use VGG16 from Simonyan and Zisserman.

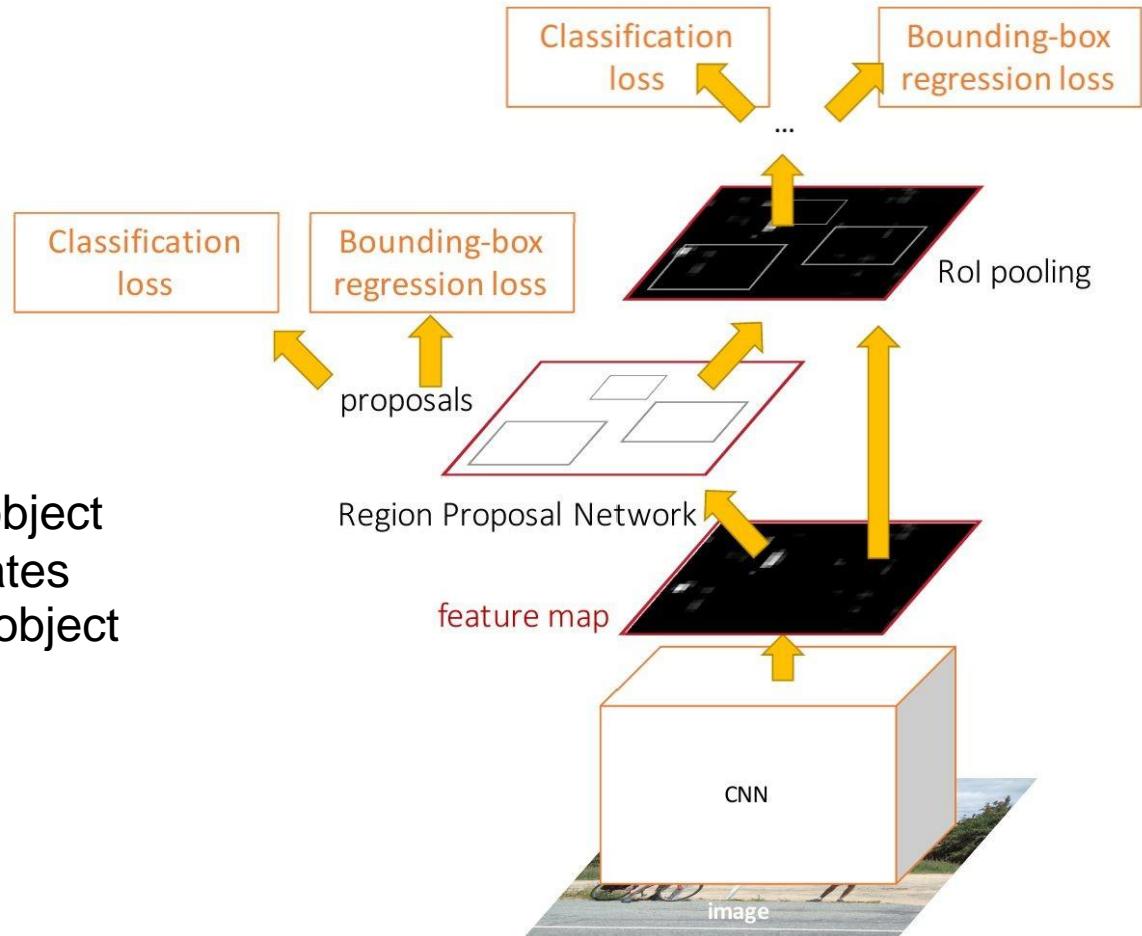
# Faster R-CNN

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



# Accurate object detection is slow!

	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img



$\frac{1}{3}$  Mile, 1760 feet



# Accurate object detection is slow!

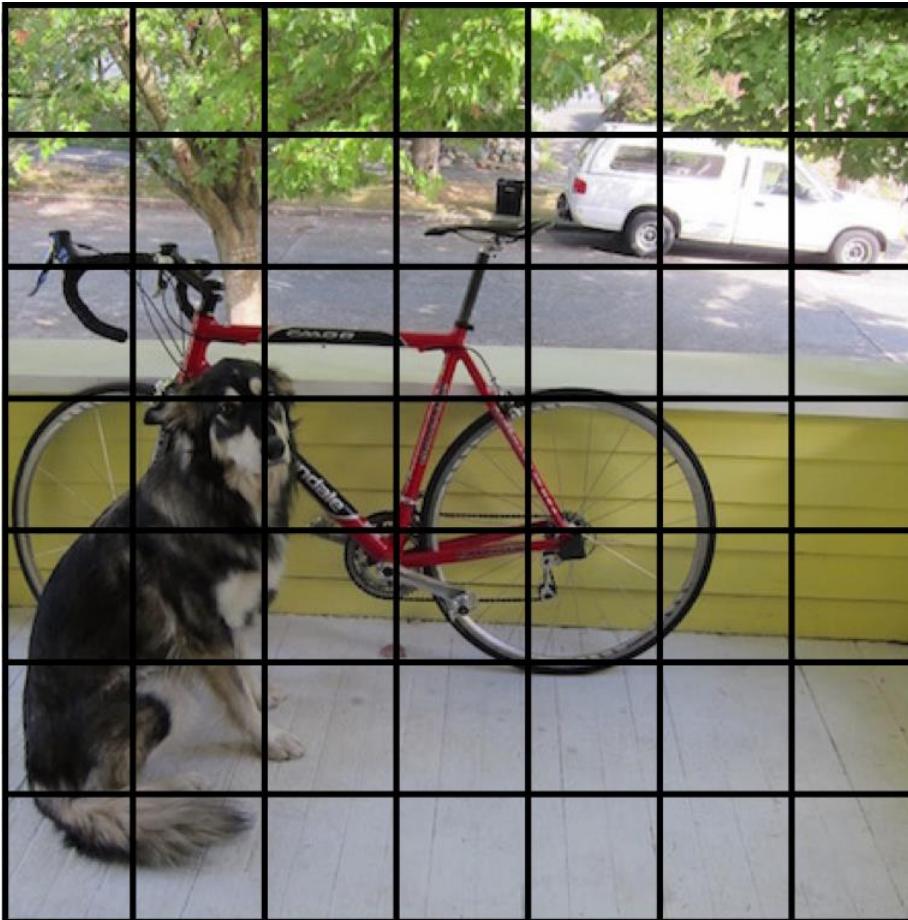
	Pascal 2007 mAP	Speed	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	69.0	45 FPS	22 ms/img



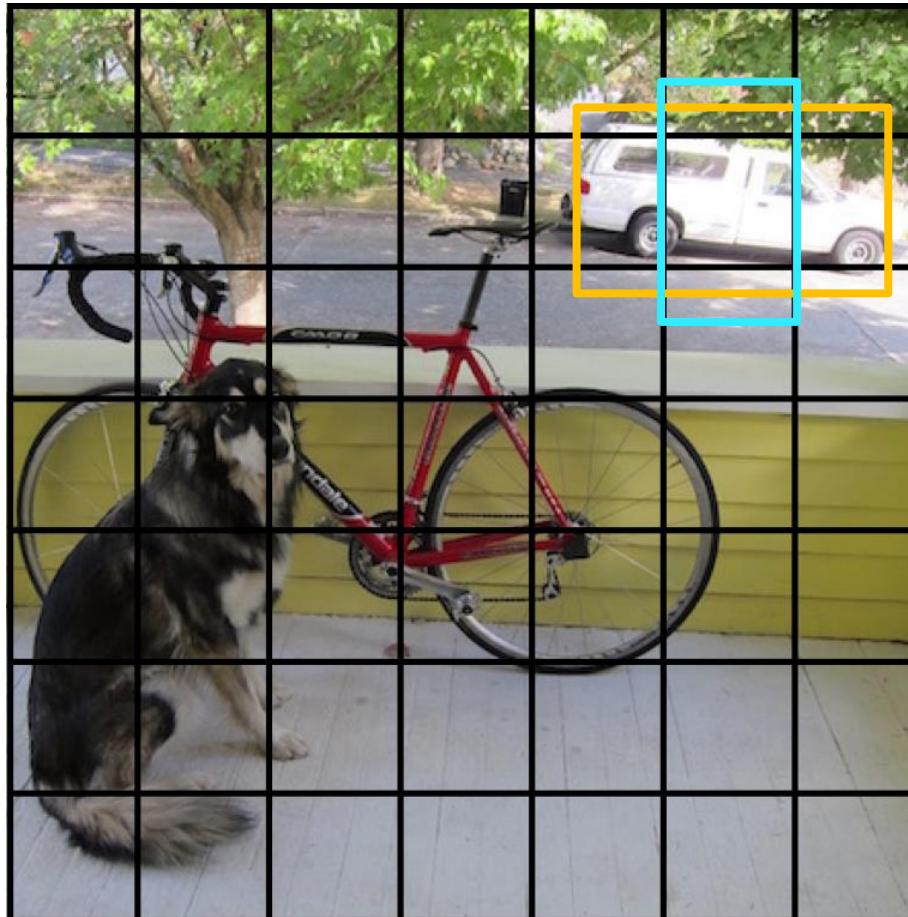
2 feet  
→

A large green arrow points to the right, with the text "2 feet" written above it in a large, bold, black font.

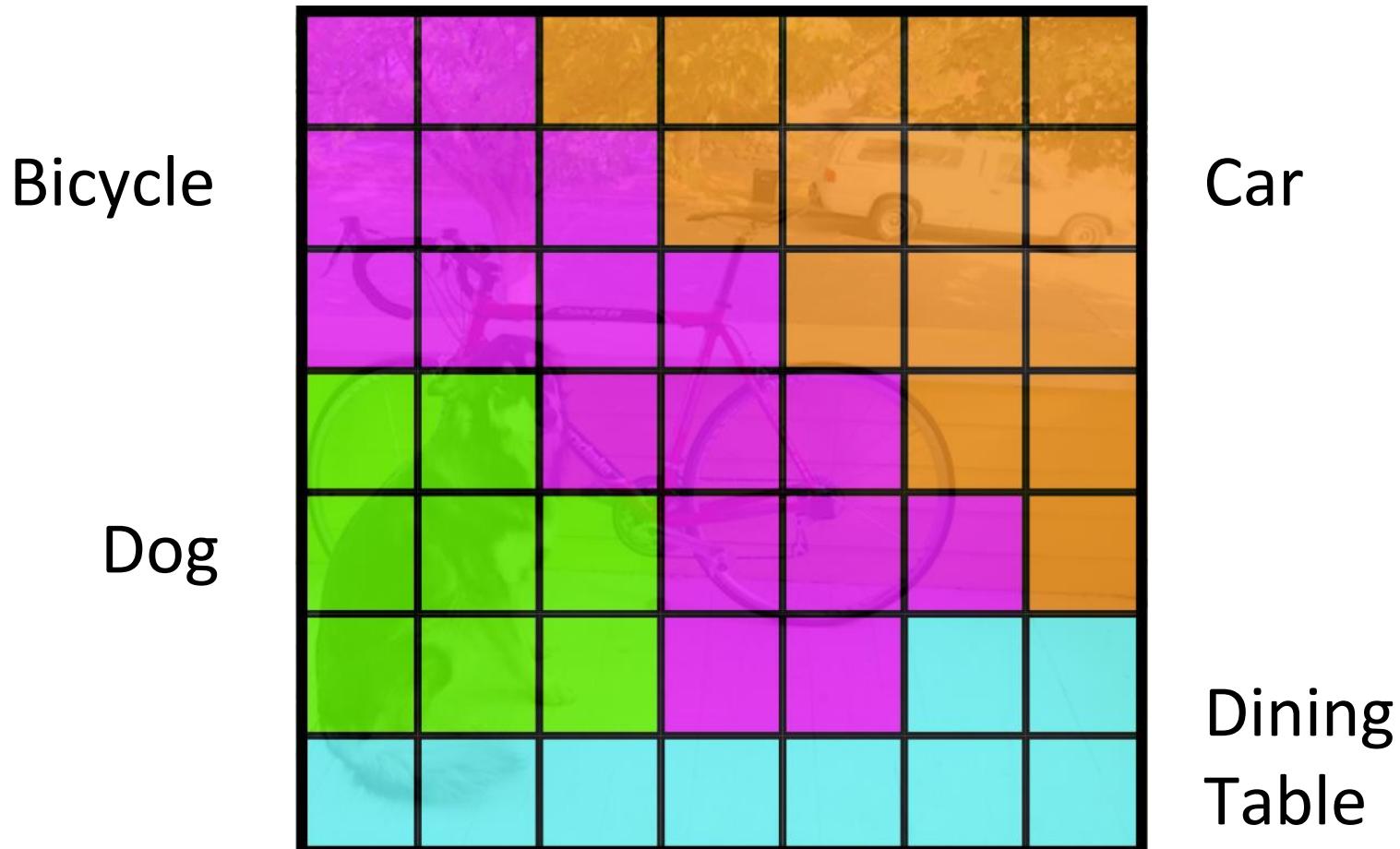
# Detection without Proposals: YOLO



Each cell predicts boxes and confidences:  
 $P(\text{Object})$  



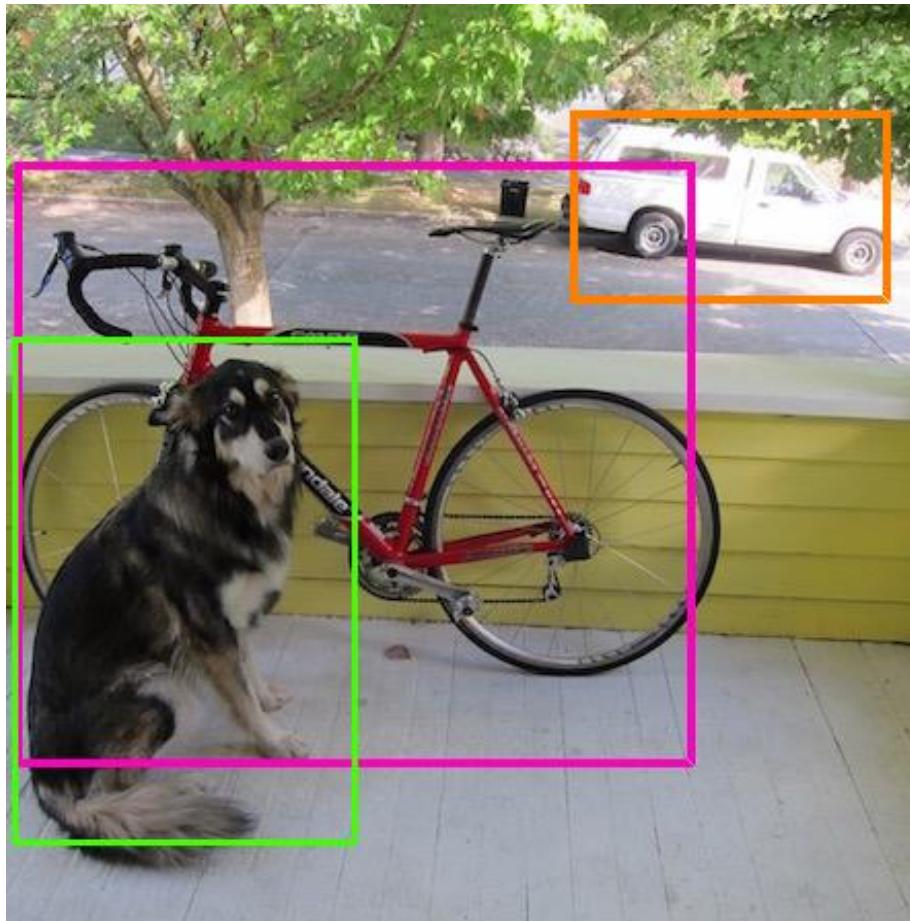
Each cell also predicts a probability  
 $P(\text{Class} \mid \text{Object})$



# Combine the box and class predictions



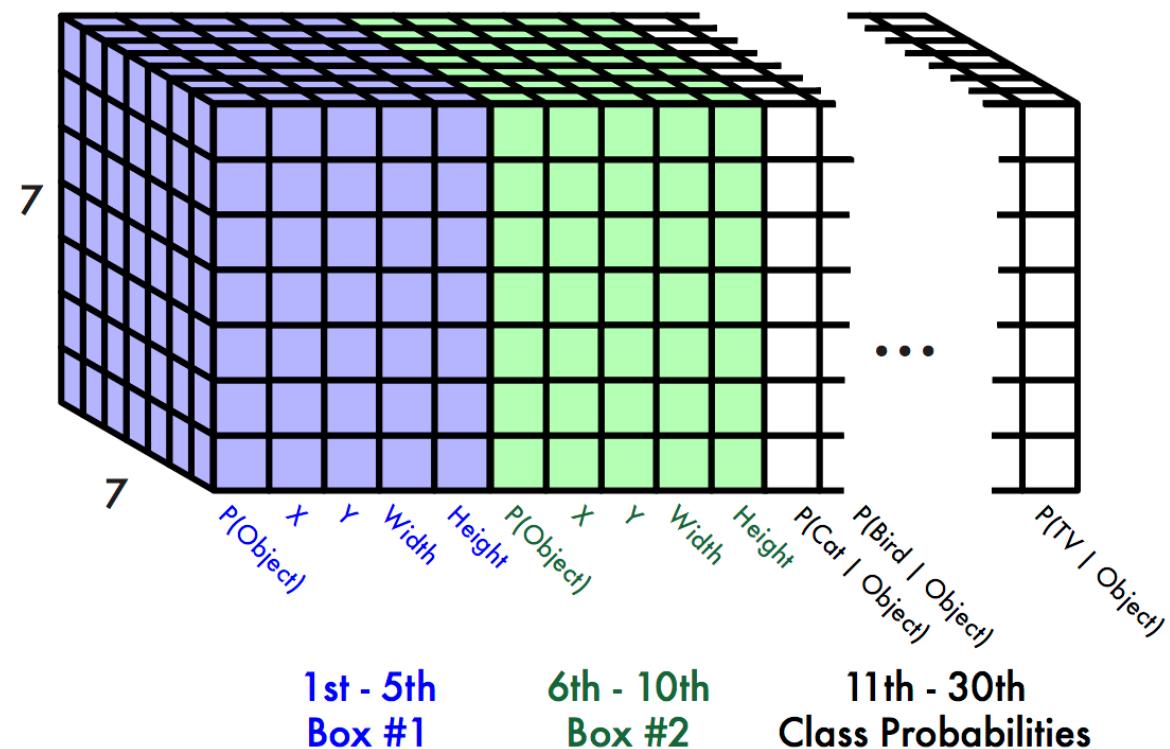
# Finally do NMS and threshold detections



# This parameterization fixes the output size

Each cell predicts:

- For each bounding box:
  - 4 coordinates ( $x, y, w, h$ )
  - 1 confidence value
- Some number of class probabilities

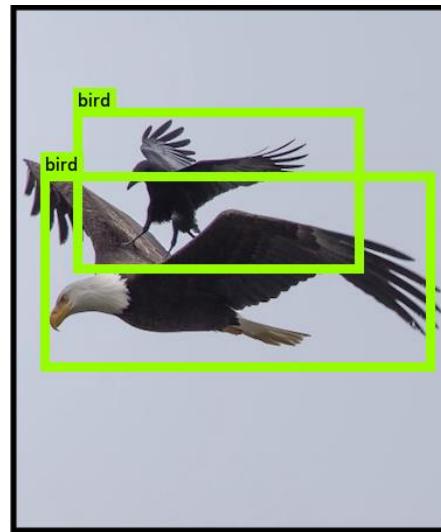
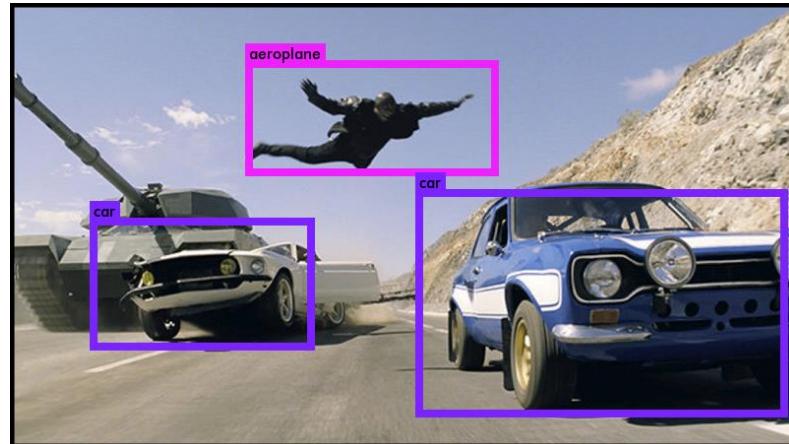
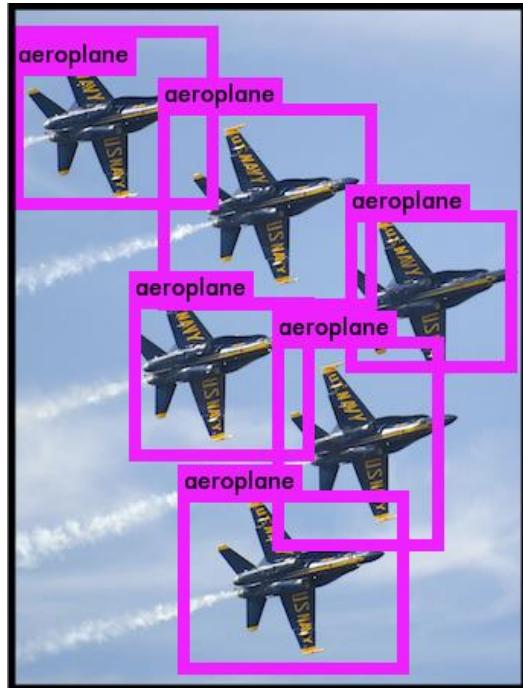


For Pascal VOC:

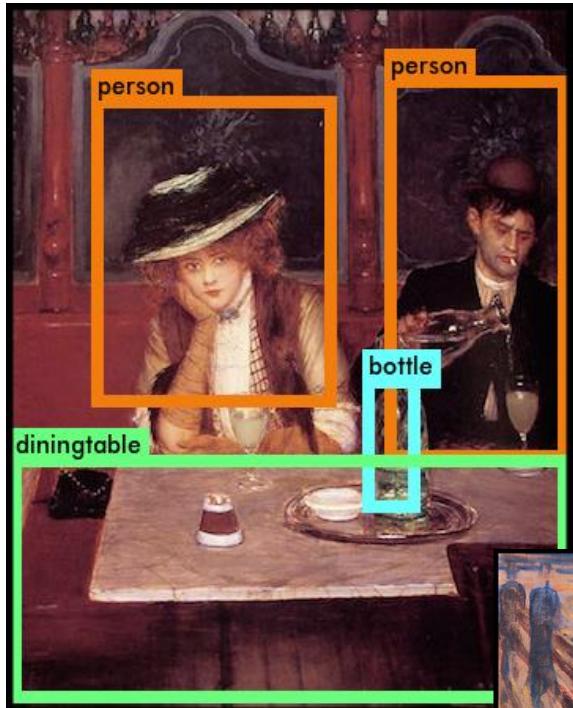
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

$$7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30 \text{ tensor} = \mathbf{1470 \text{ outputs}}$$

# YOLO works across many natural images



# It also generalizes well to new domains



JOSEPH            ALI  
REDMON        FARHADI

RETURN IN.....

# YOLO9000

## Better, Faster, Stronger

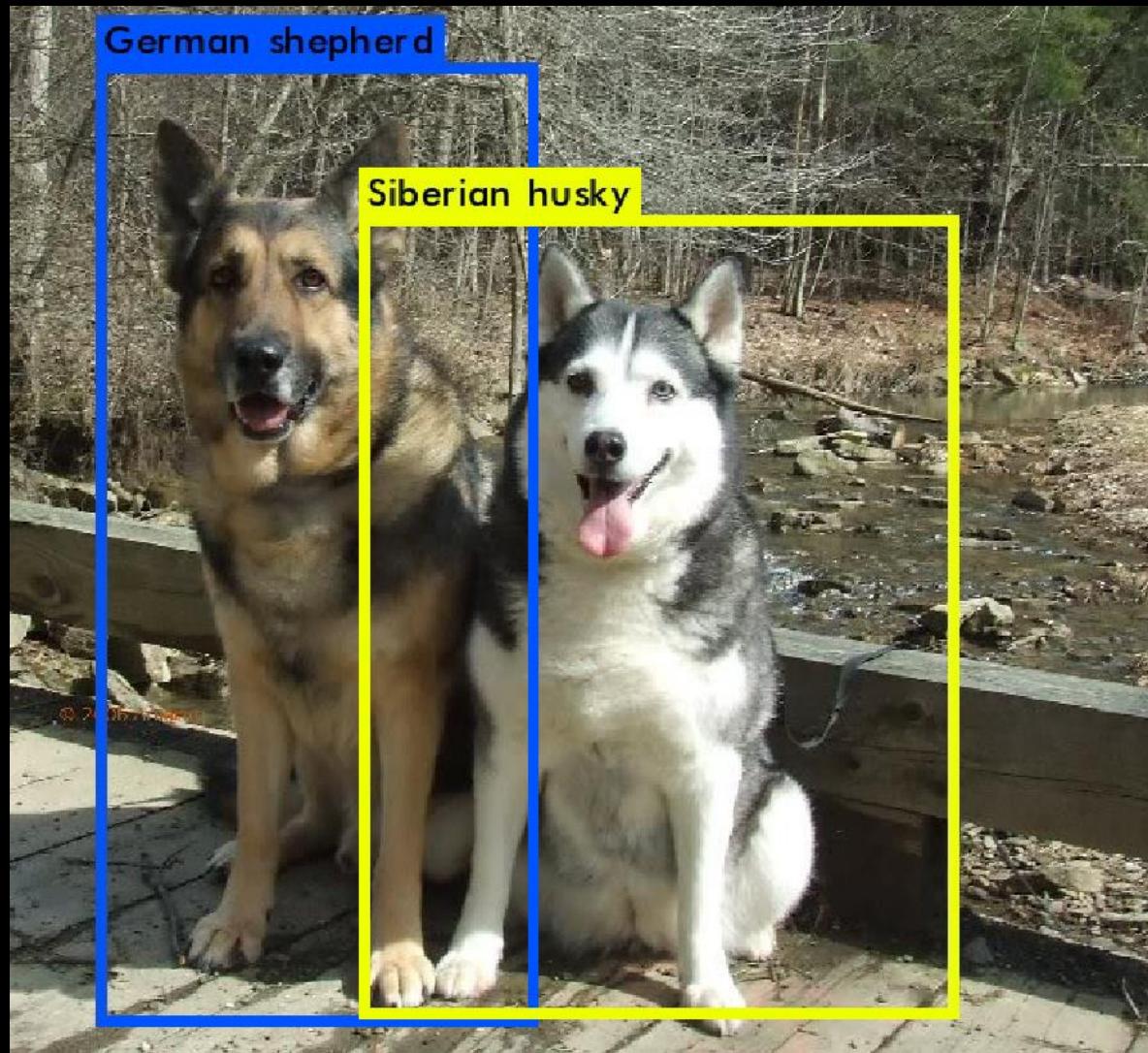
NOW PLAYING IN A DEMO NEAR YOU

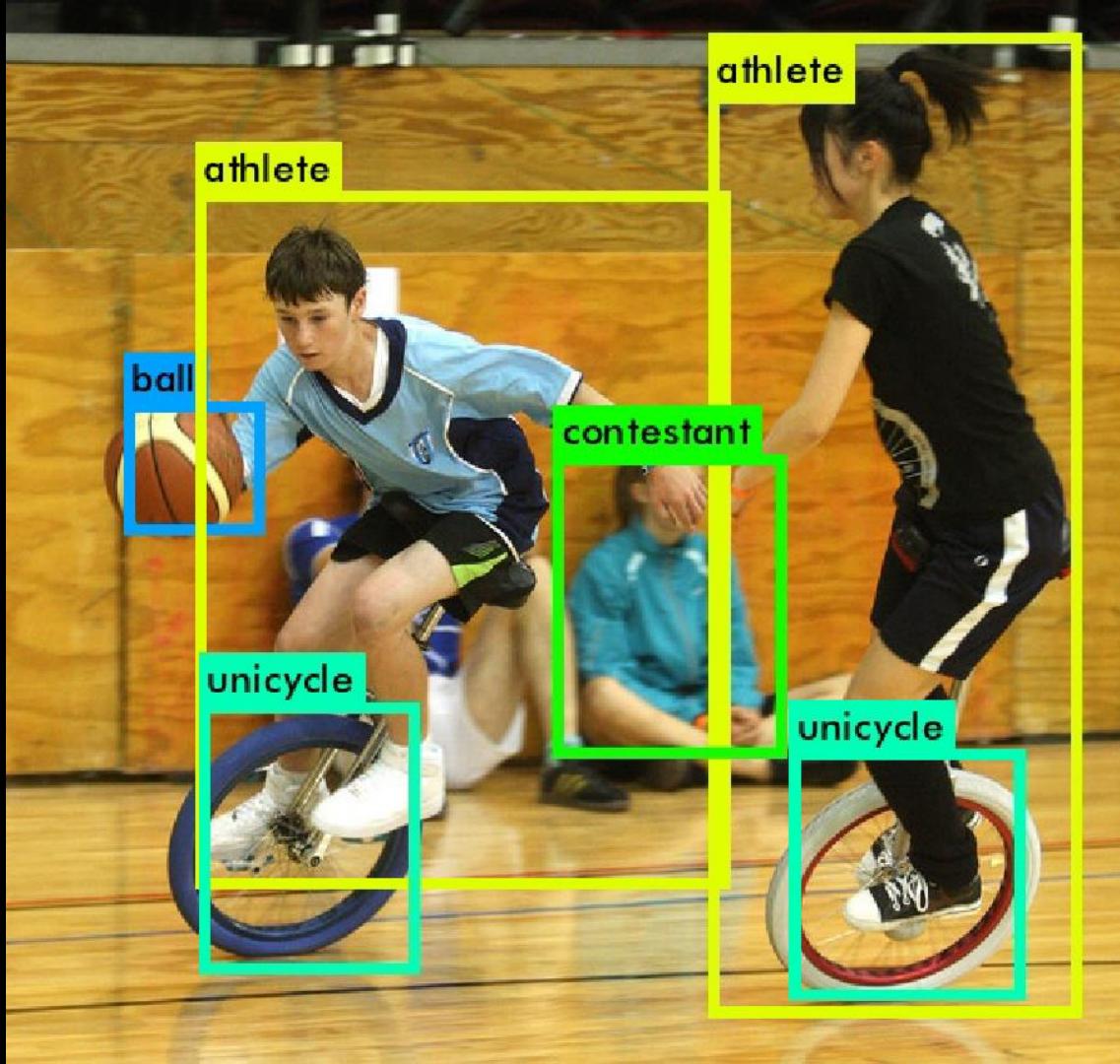
UNIVERSITY OF WASHINGTON PRESENTED IN ASSOCIATION WITH XNOR.AI AND THE ALLEN INSTITUTE FOR ARTIFICIAL INTELLIGENCE  
MODELS BY DARKNET; OPEN SOURCE NEURAL NETWORKS

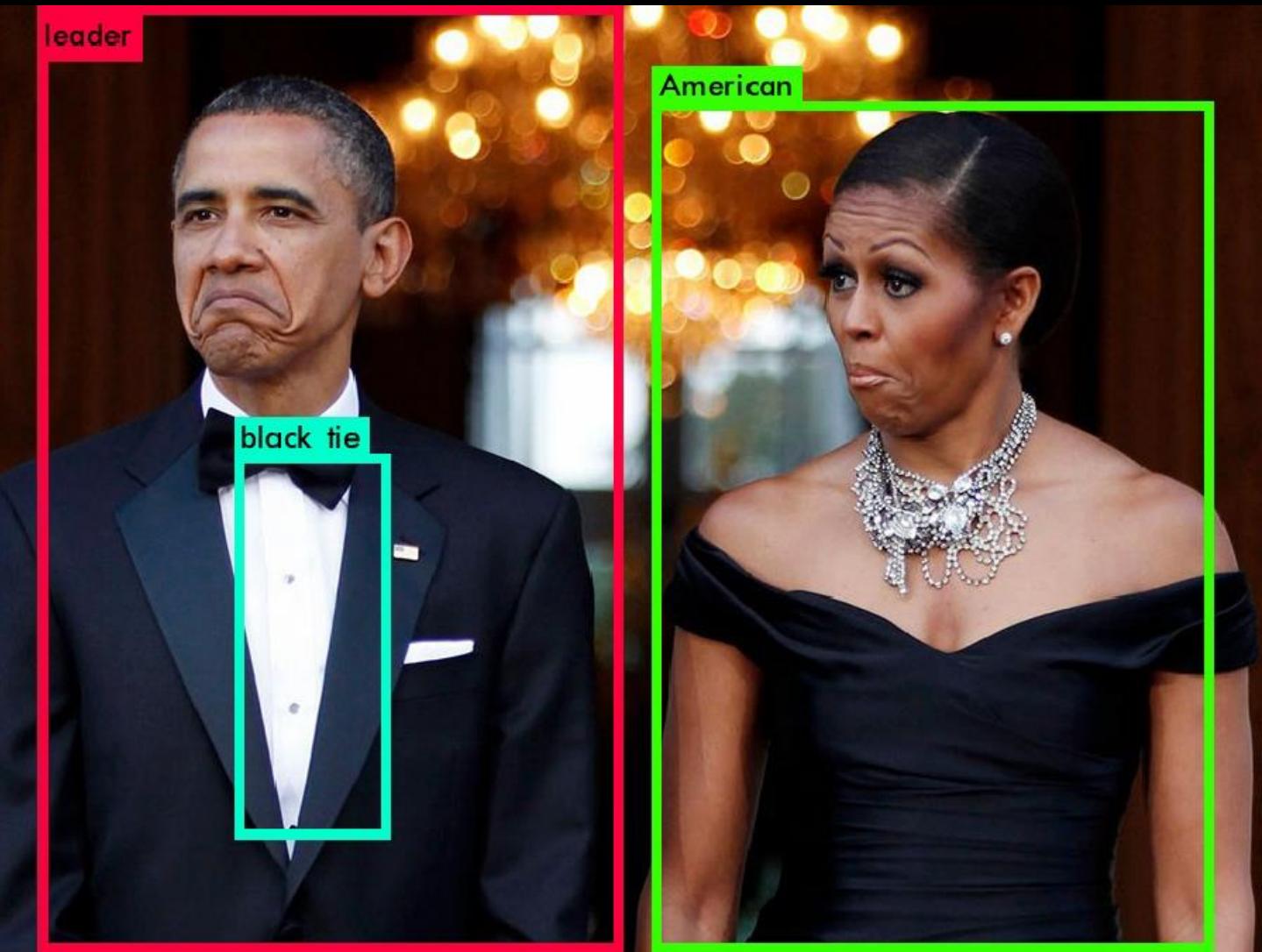
@DARKNETFOREVER #YOLO9000

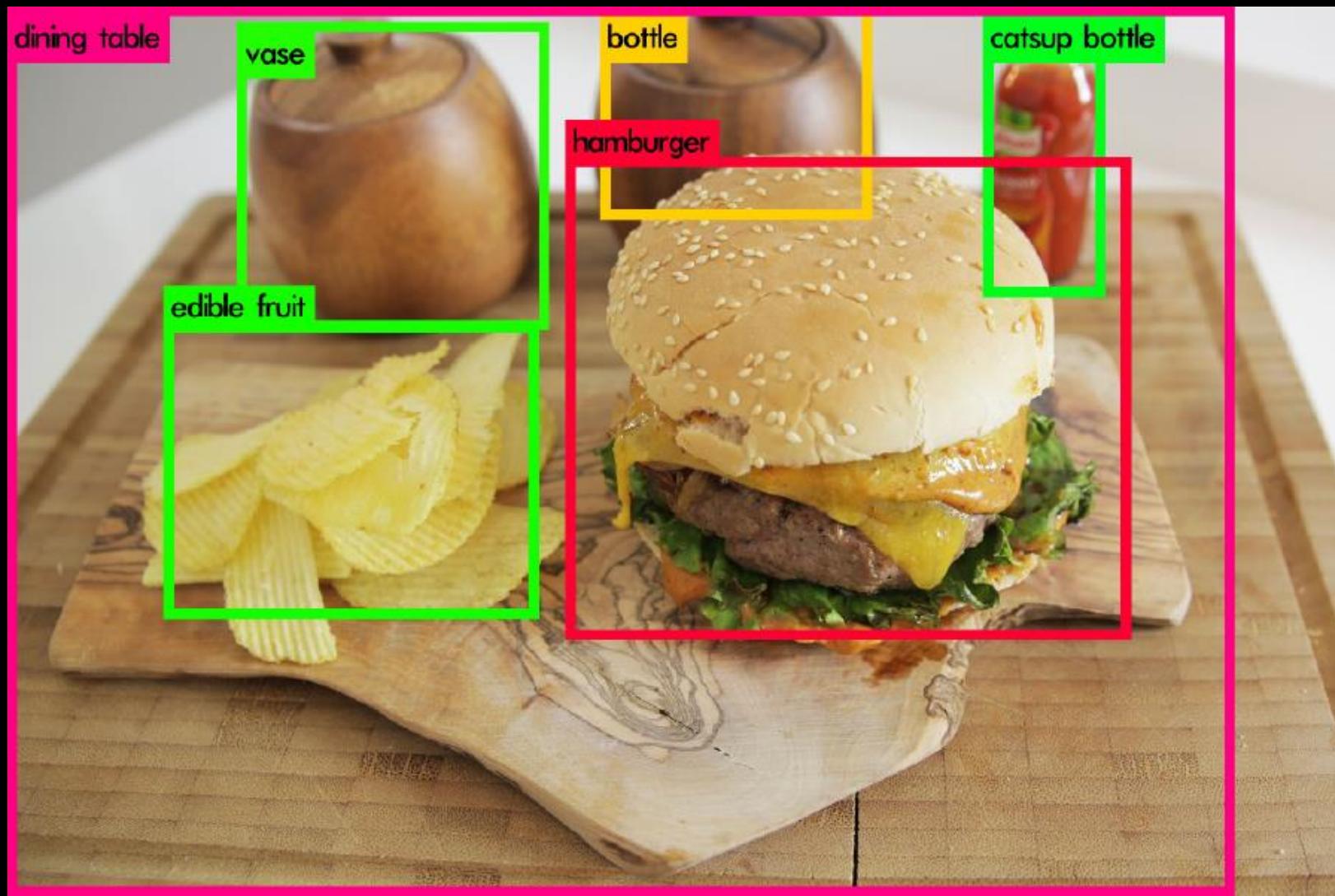
[pjreddie.com/yolo](http://pjreddie.com/yolo)













# Plan for the next two lectures

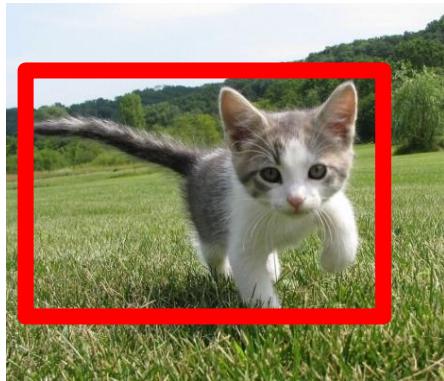
- Detection approaches
  - Pre-CNNs
    - Detection with whole windows: Pedestrian detection
    - Part-based detection: Deformable Part Models
  - Post-CNNs
    - Detection with region proposals: R-CNN, Fast R-CNN, Faster-R-CNN
    - Detection without region proposals: YOLO, SSD
- Segmentation approaches
  - Semantic segmentation: FCN
  - Instance segmentation: Mask R-CNN

# Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels



CAT

Single Object



DOG, DOG, CAT

Multiple Object

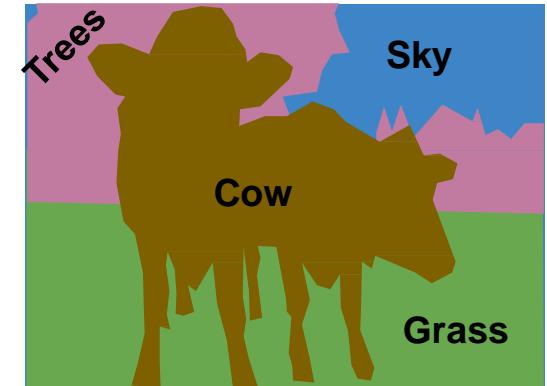
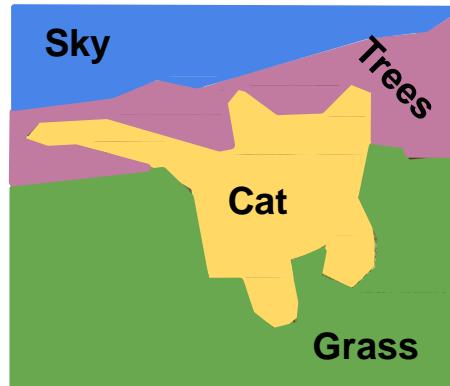


DOG, DOG, CAT

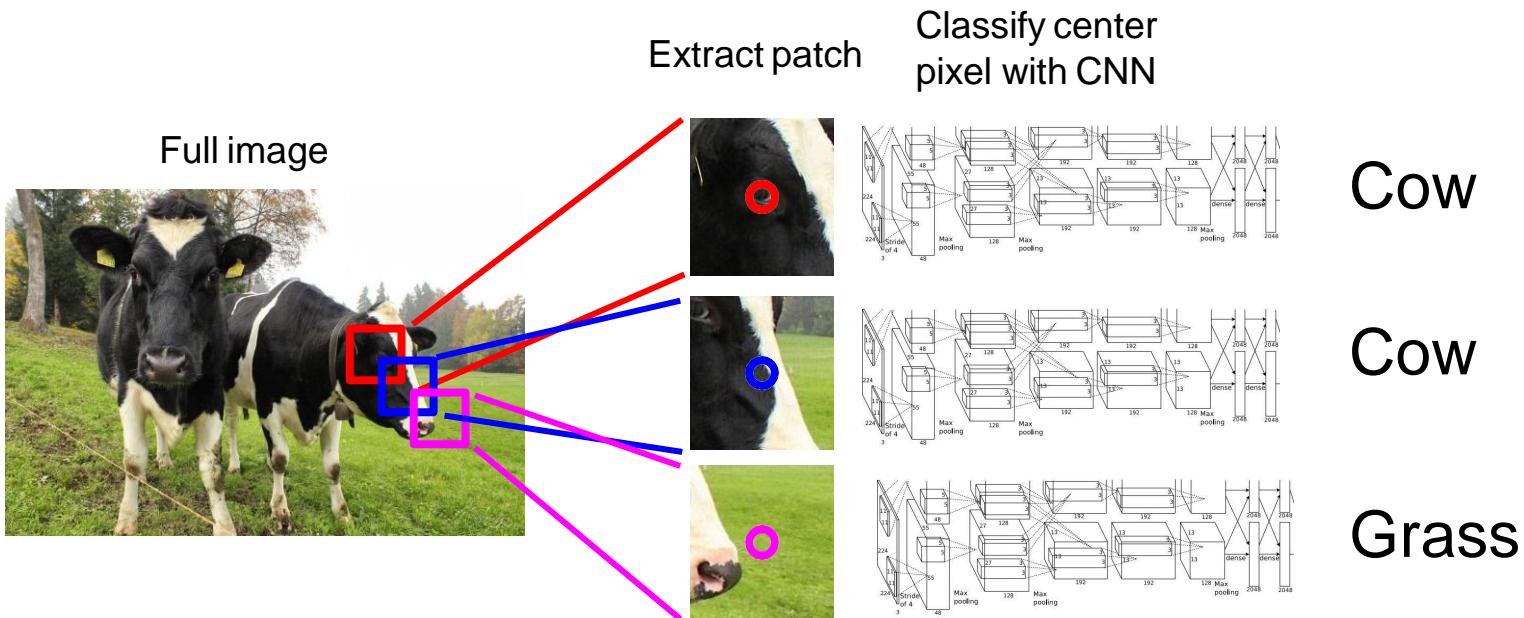
# Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels

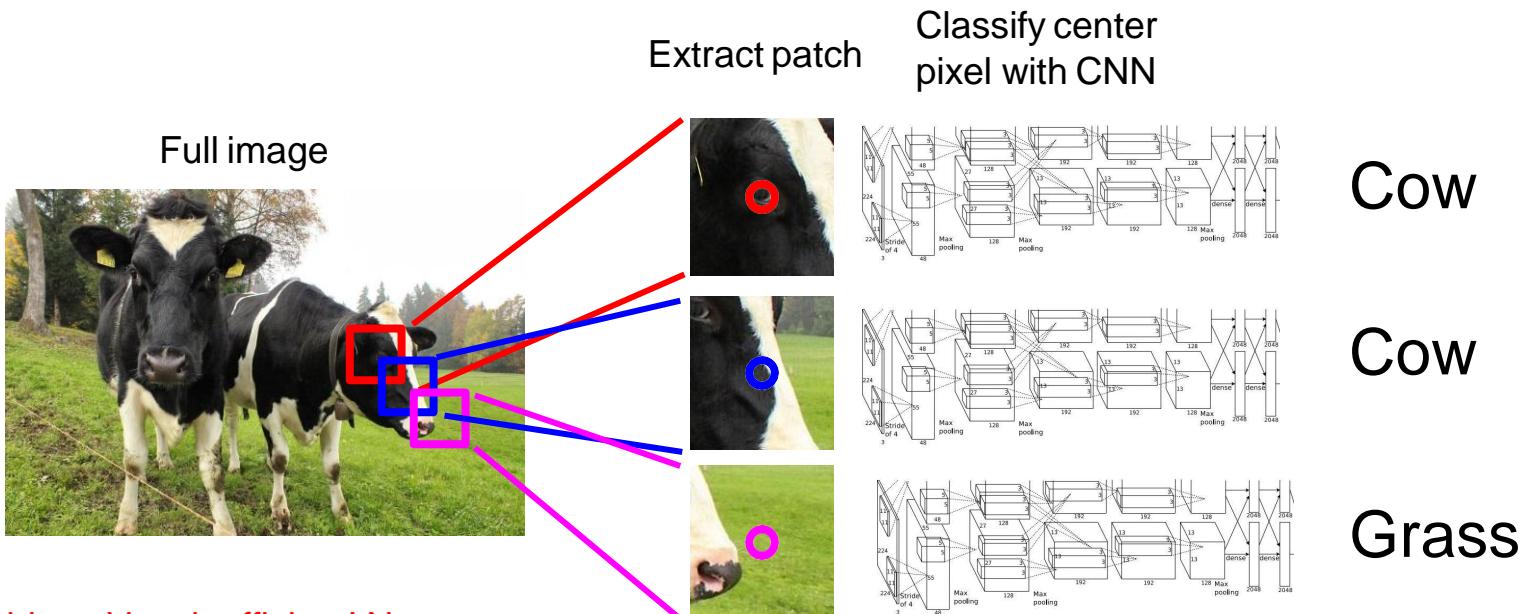


# Semantic Segmentation Idea: Sliding Window



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

# Semantic Segmentation Idea: Sliding Window

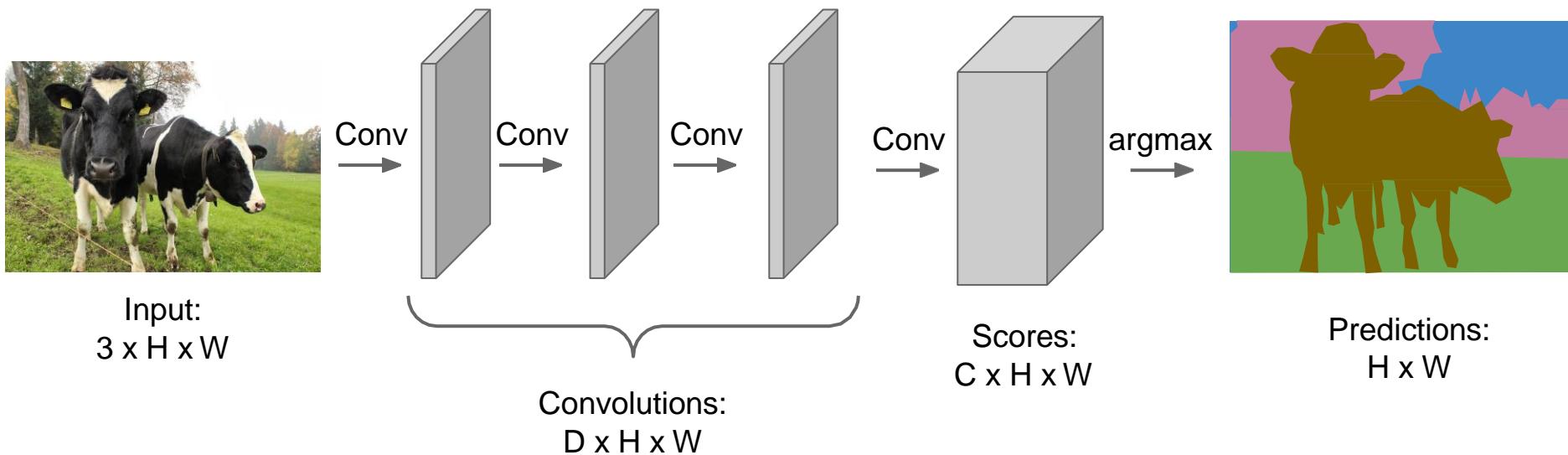


Problem: Very inefficient! Not  
reusing shared features between  
overlapping patches

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

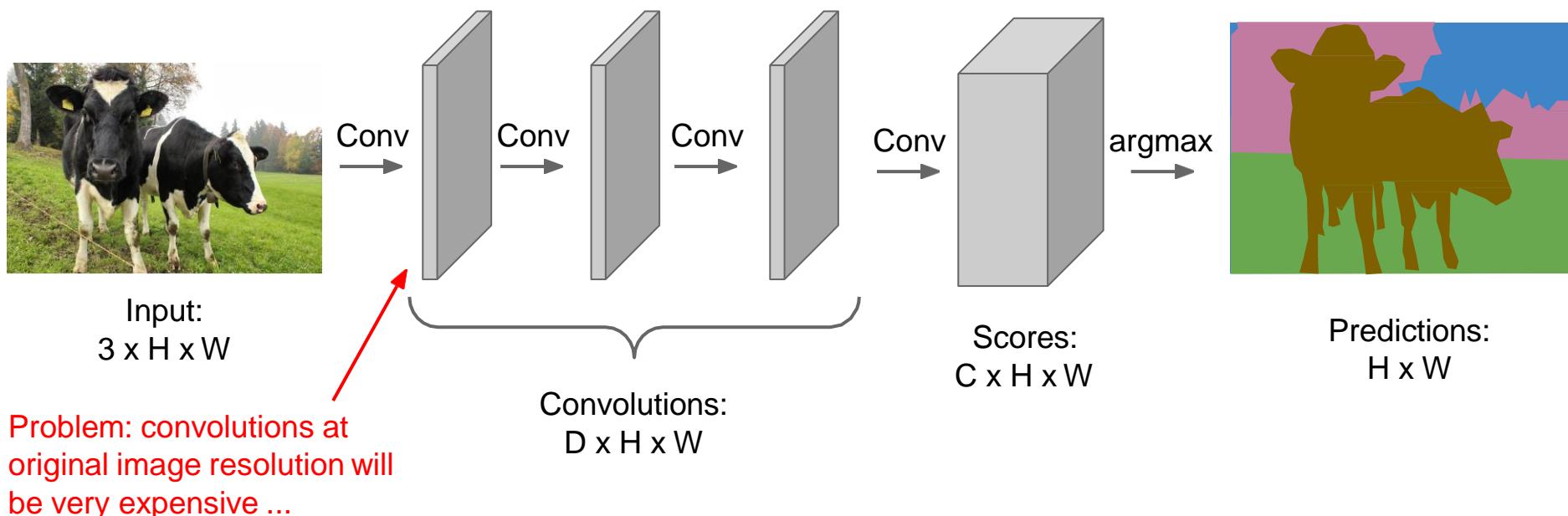
# Semantic Segmentation Idea: Fully Convolutional

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



# Semantic Segmentation Idea: Fully Convolutional

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

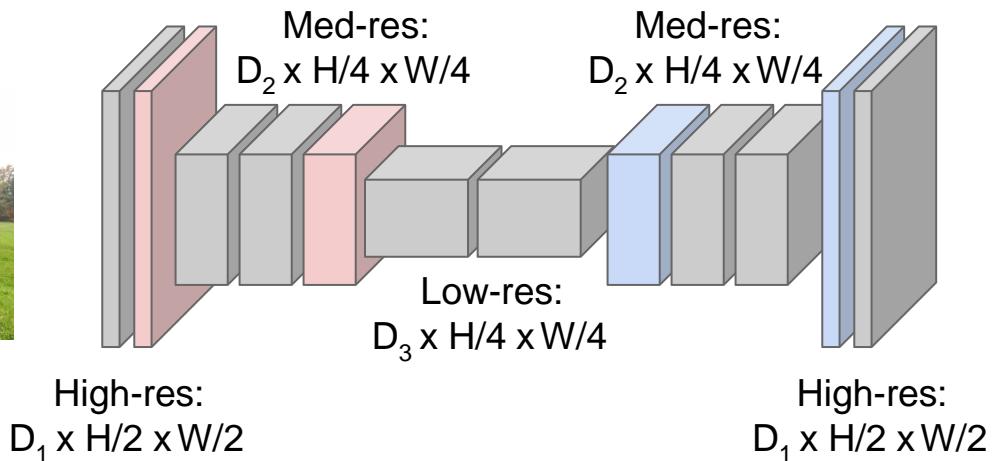


# Semantic Segmentation Idea: Fully Convolutional

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

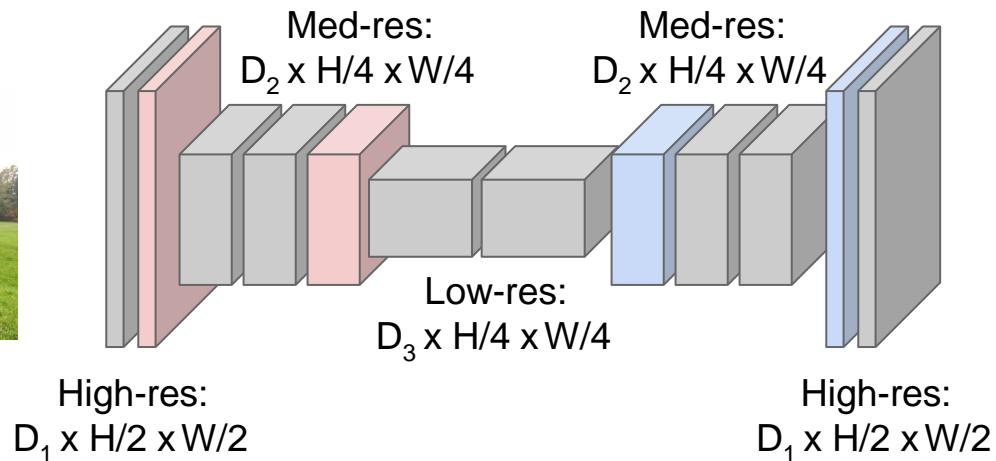
# Semantic Segmentation Idea: Fully Convolutional

**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

# 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

# In-Network upsampling: “Max Unpooling”

## Max Pooling

Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

## Max Unpooling

Use positions from pooling layer

1	2
3	4

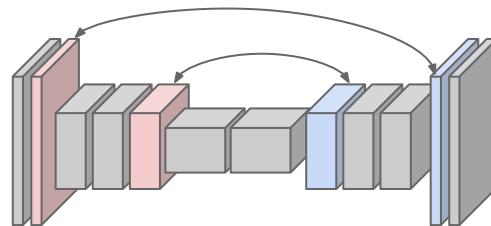
...

Rest of the network

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

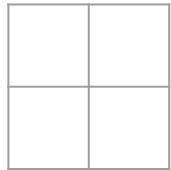
Output: 4 x 4

Corresponding pairs of  
downsampling and  
upsampling layers

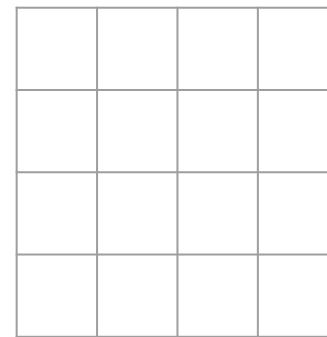


# Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



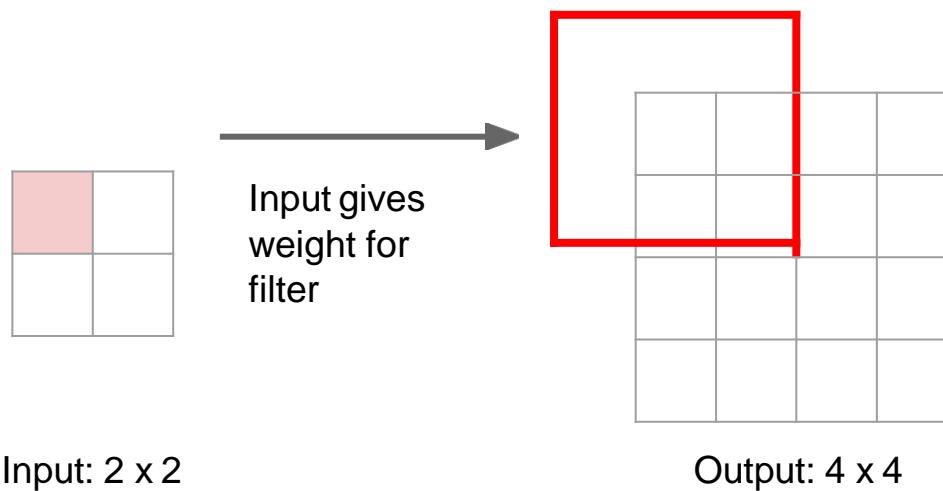
Input: 2 x 2



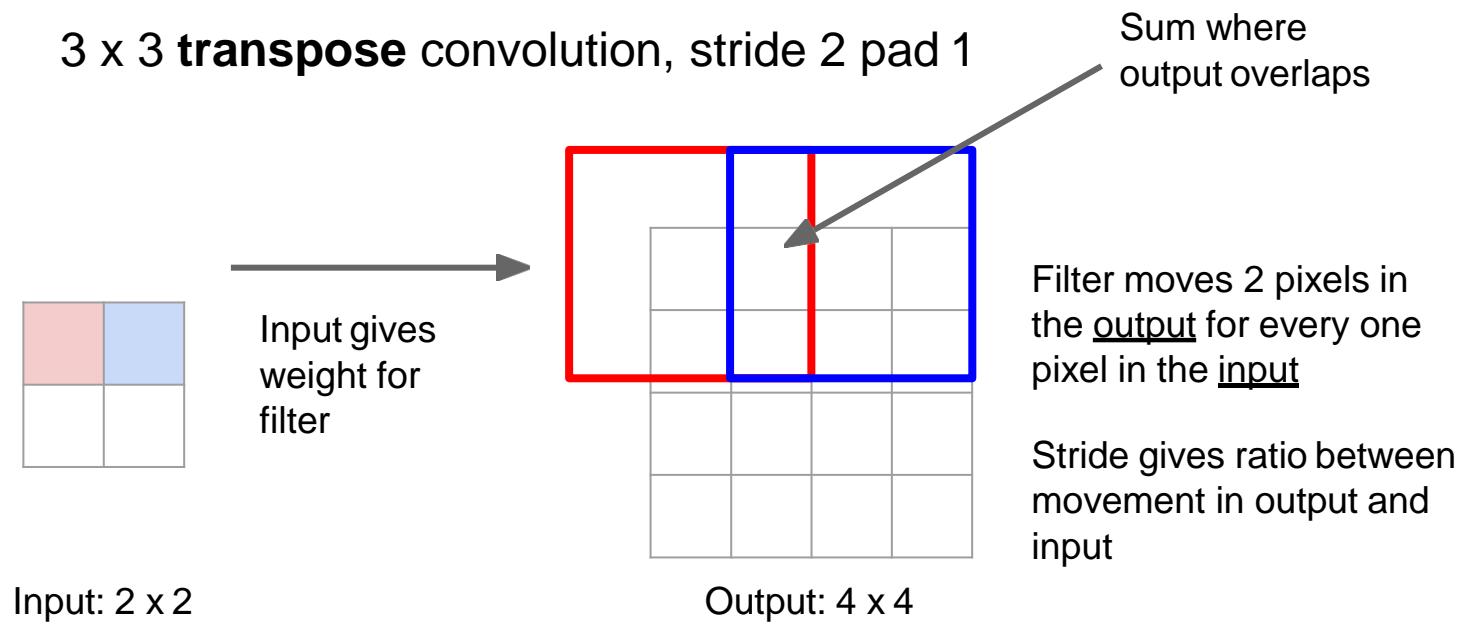
Output: 4 x 4

# Learnable Upsampling: Transpose Convolution

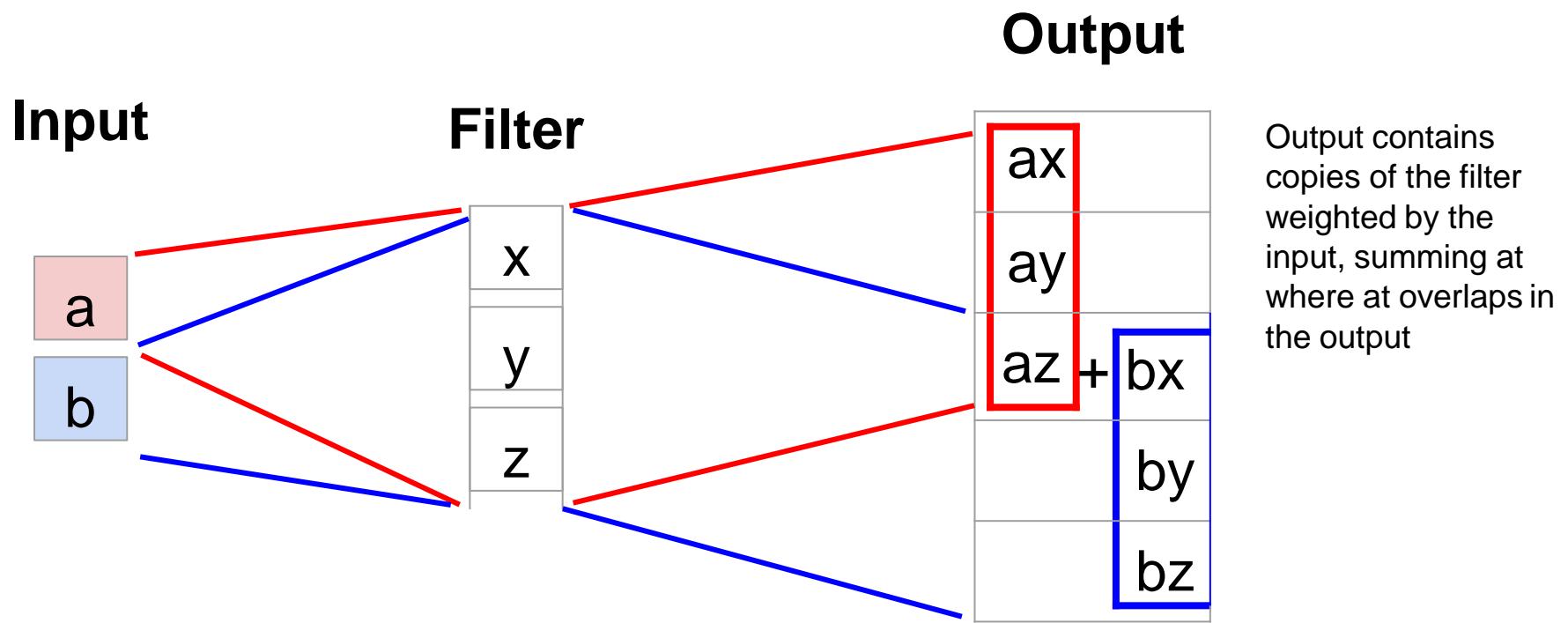
3 x 3 **transpose** convolution, stride 2 pad 1



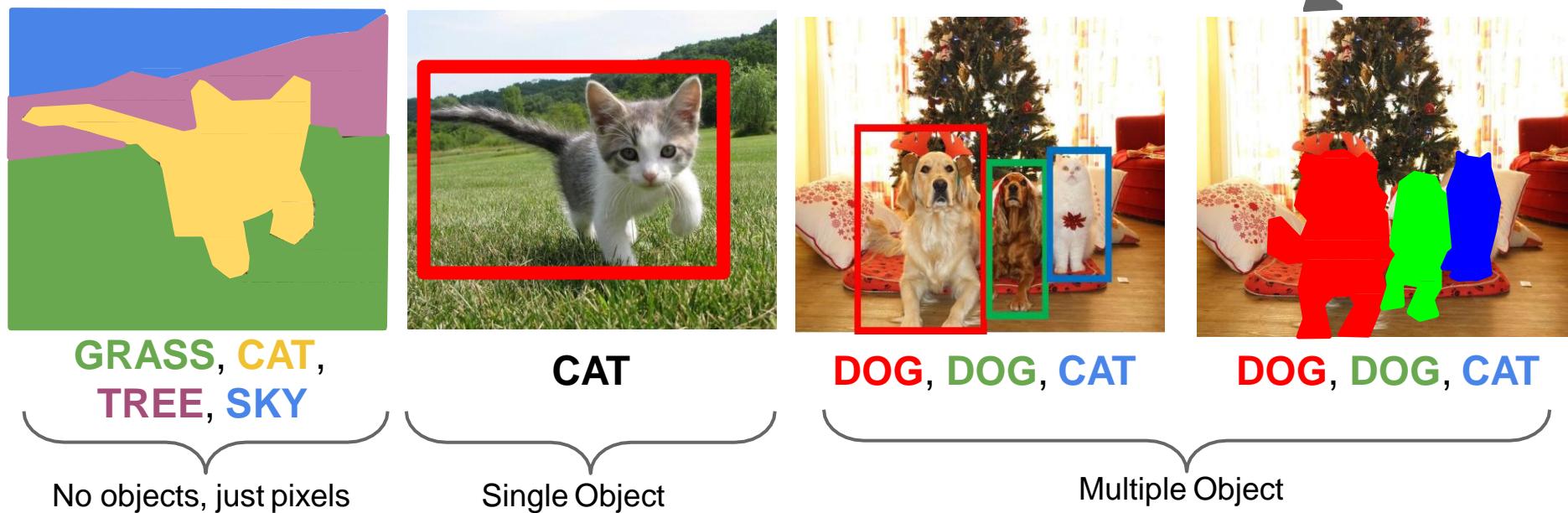
# Learnable Upsampling: Transpose Convolution



# Transpose Convolution: 1D Example



# Instance Segmentation

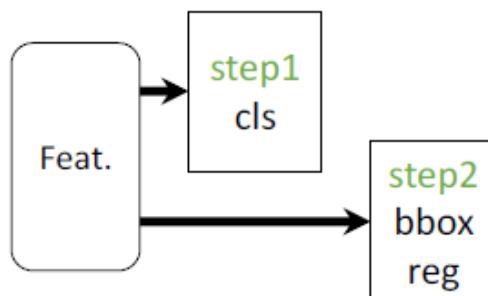


# Mask R-CNN

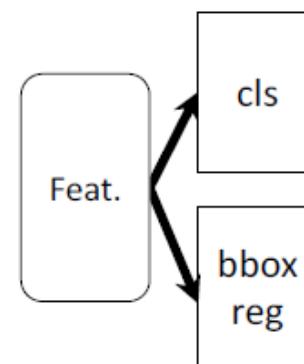
He et al, "Mask R-CNN", ICCV 2017

## What is Mask R-CNN: Parallel Heads

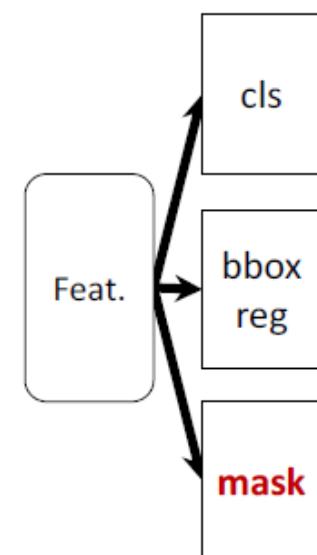
- Easy, fast to implement and use



(slow) R-CNN



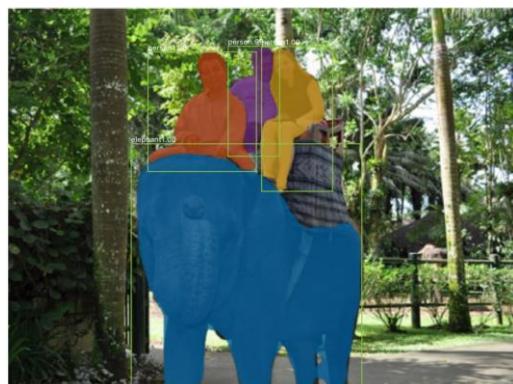
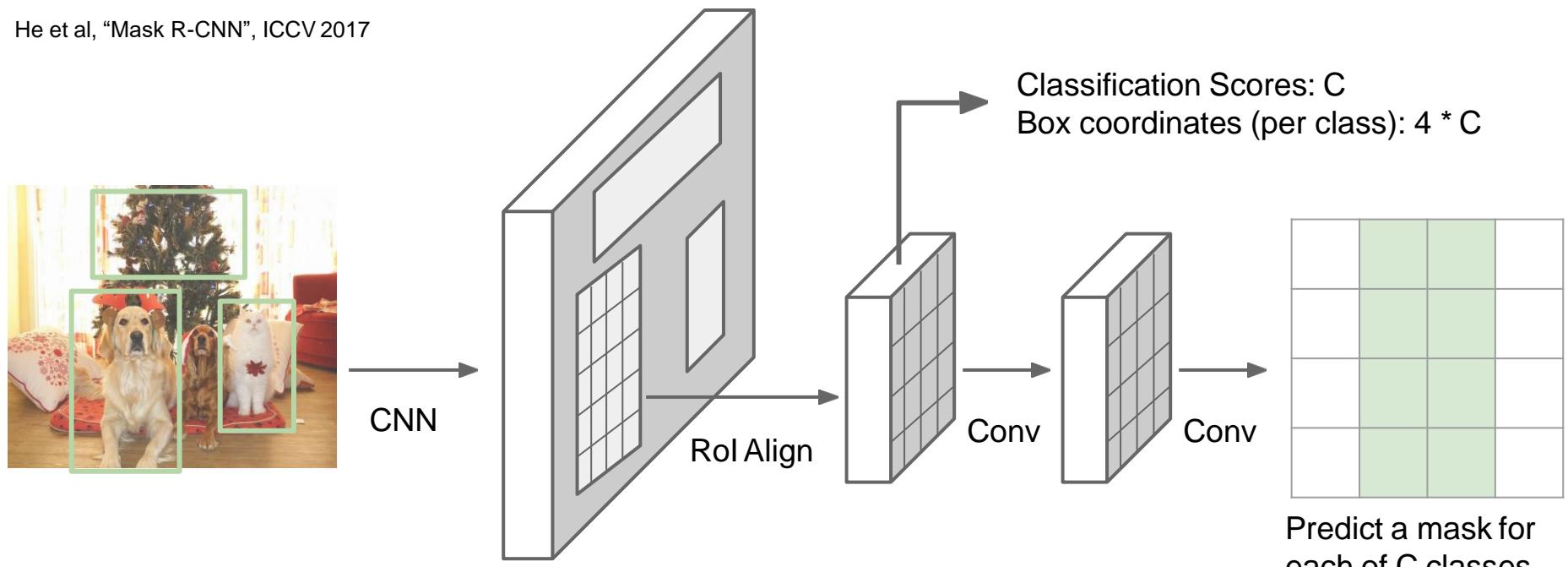
Fast/er R-CNN



Mask R-CNN

# Mask R-CNN

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



Adapted from Justin Johnson