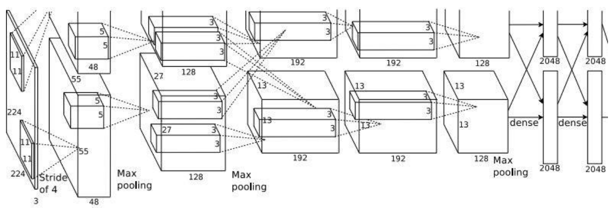
CS 2770: Computer Vision

Object Detection

Prof. Adriana Kovashka
University of Pittsburgh
March 3, 2020

So far: Image Classification



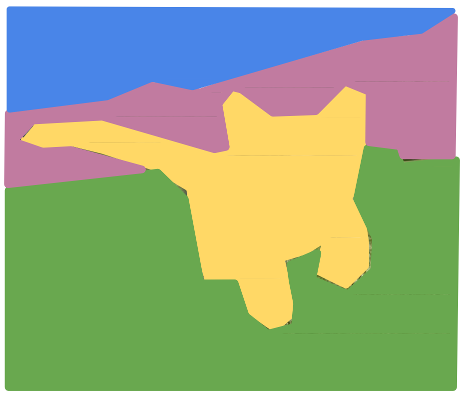
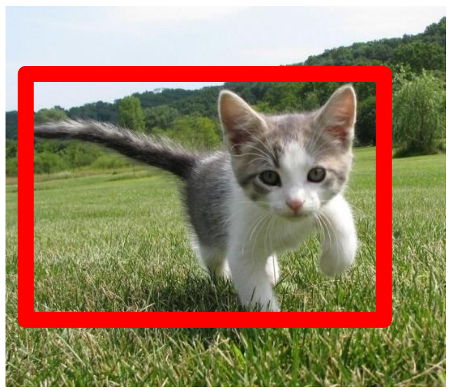
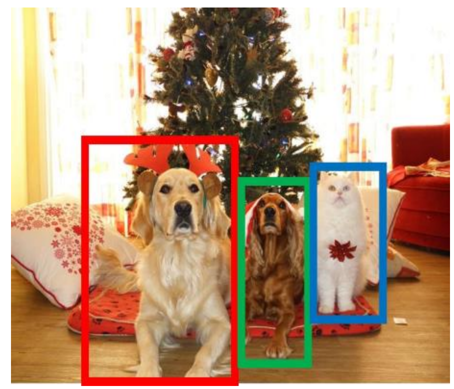
Vector:
4096

Fully-Connected:
4096 to 1000

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

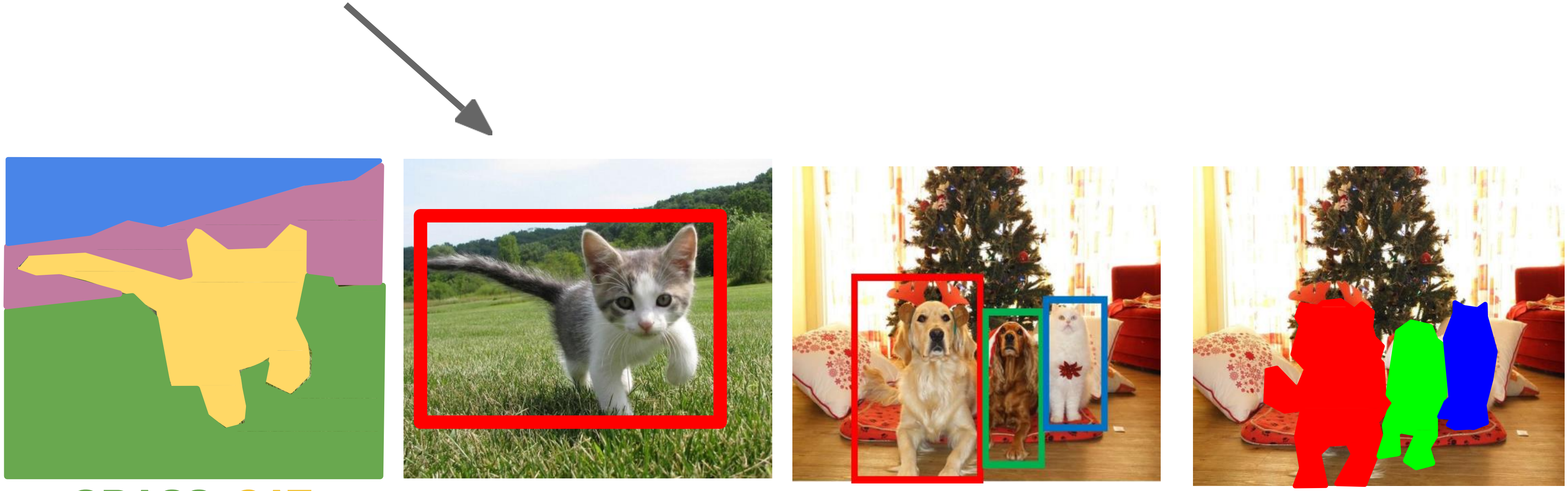
Slide by: Justin Johnson

Other Computer Vision Tasks

Semantic Segmentation	**Classification + Localization**	**Object Detection**	**Instance Segmentation**
			
GRASS, CAT, TREE, SKY	CAT	DOG, DOG, CAT	DOG, DOG, CAT
No objects, just pixels	Single Object	Multiple Object	

Slide by: Justin Johnson

Classification + Localization



GRASS, CAT, TREE, SKY

CAT

DOG, DOG, CAT

DOG, DOG, CAT

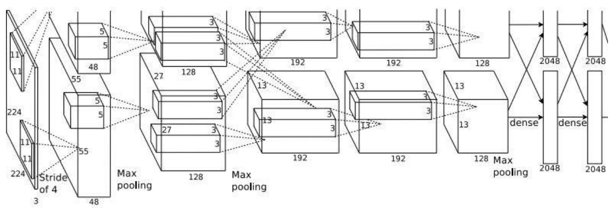
No objects, just pixels

Single Object

Multiple Object

Slide by: Justin Johnson

Classification + Localization



Fully Connected:
4096 to 1000

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

Vector:
4096

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

Treat localization as a regression problem!

Slide by: Justin Johnson

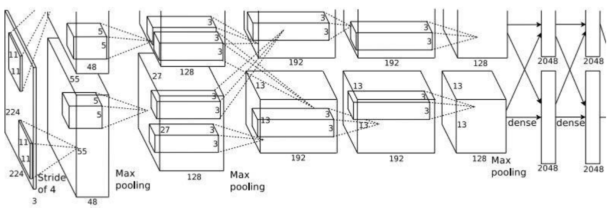
Classification + Localization

Correct label:
Cat

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

Softmax Loss

Fully Connected:
4096 to 1000



Vector:
4096

Fully Connected:
4096 to 4

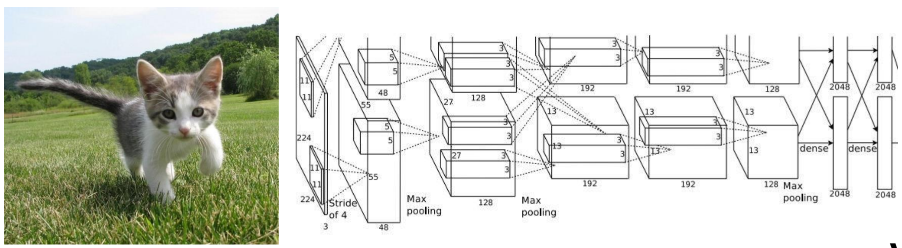
Box Coordinates
(x, y, w, h)

L2 Loss

Correct box:
(x’, y’, w’, h’)

Treat localization as a regression problem!

Classification + Localization



Vector: 4096

Fully Connected: 4096 to 1000

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

Correct label:
Cat

Softmax Loss

Multitask Loss

+ → **Loss**

Fully Connected: 4096 to 4

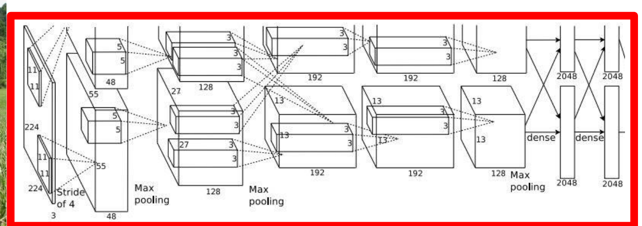
Box Coordinates
(x, y, w, h)

L2 Loss

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

Treat localization as a regression problem!

Classification + Localization



Often pretrained on ImageNet
(Transfer learning)

Vector:
4096

Fully Connected:
4096 to 1000

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

Correct label:
Cat

Softmax Loss

+

Loss

Fully Connected:
4096 to 4

Box Coordinates
(x, y, w, h)

L2 Loss

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

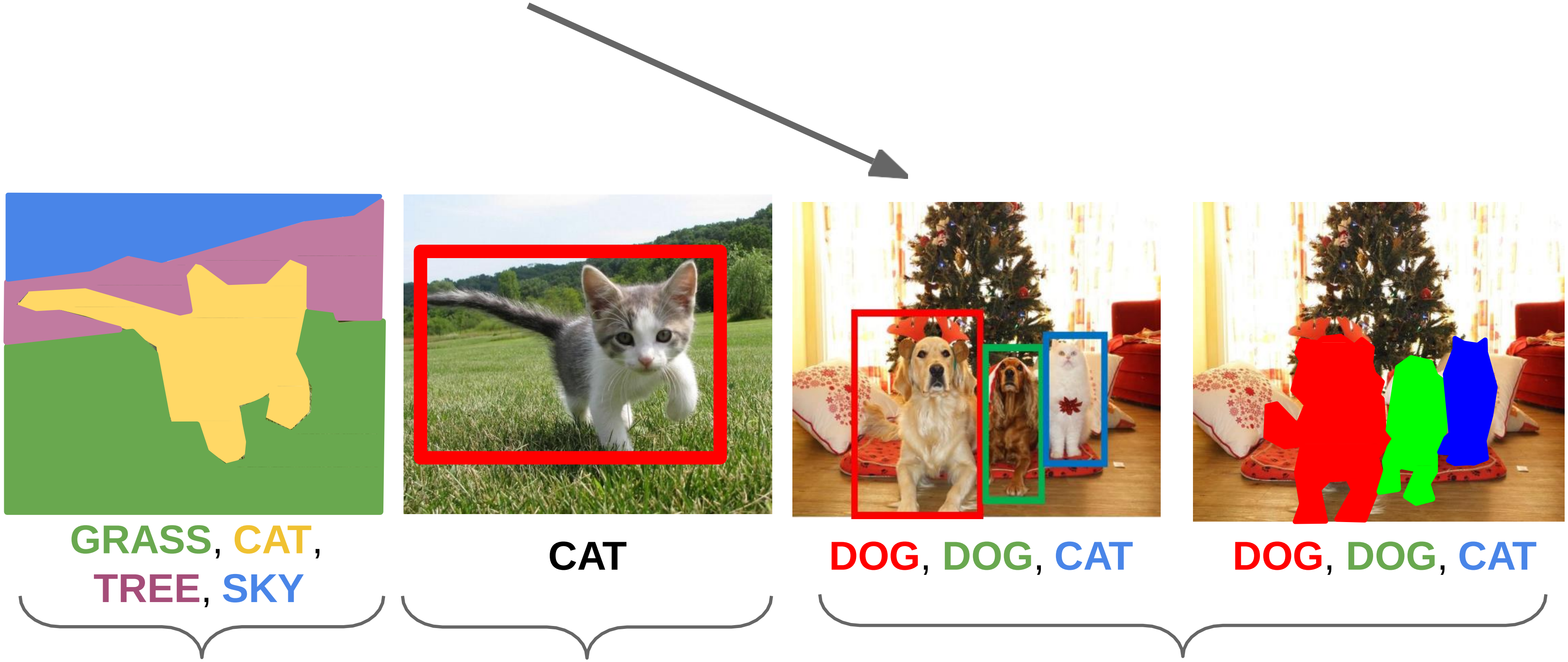
Treat localization as a regression problem!

Slide by: Justin Johnson

Plan for this lecture

- Fully supervised detection
  - Pre-CNN: Deformable part models
  - Detection with region proposals: R-CNN, Fast/er R-CNN
  - Detection without region proposals: YOLO
  - Semantic and instance segmentation: FCN, Mask R-CNN
- Weak or out-of-domain supervision
  - Weakly supervised object detection
  - Domain adaptation

Object Detection



No objects, just pixels

Single Object

Multiple Object

Slide by: Justin Johnson

Object detection: basic framework

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

Adapted from Kristen Grauman

Window-template-based models
Building an object model

Given the representation, train a binary classifier

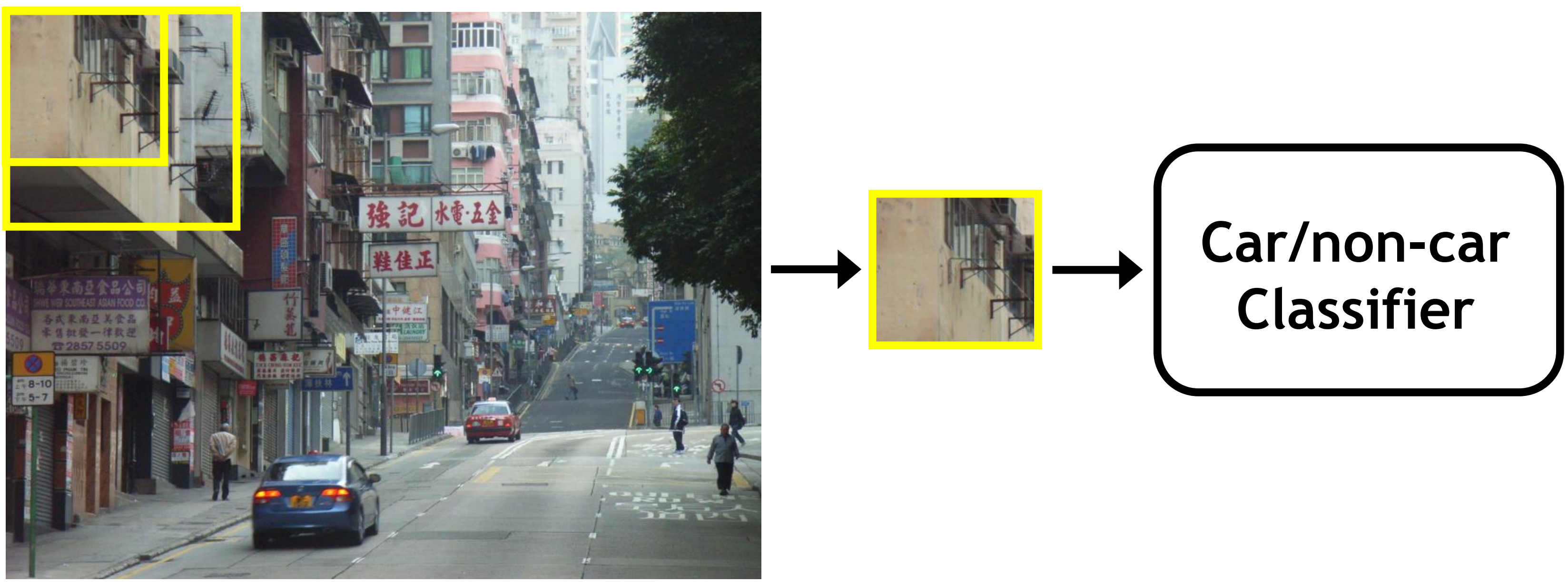
Car/non-car
Classifier

No, not a car.

Yes, car.

Kristen Grauman

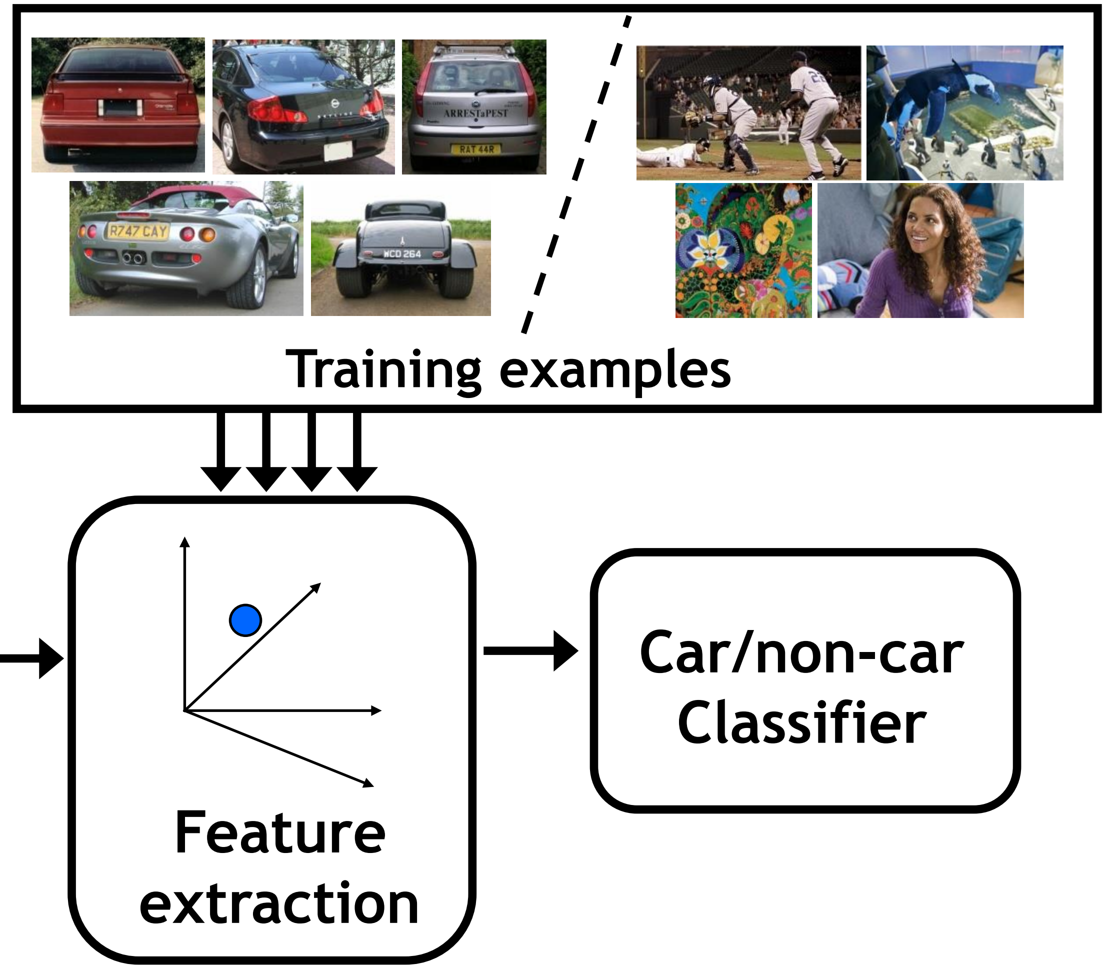
Window-template-based models
Generating and scoring candidates

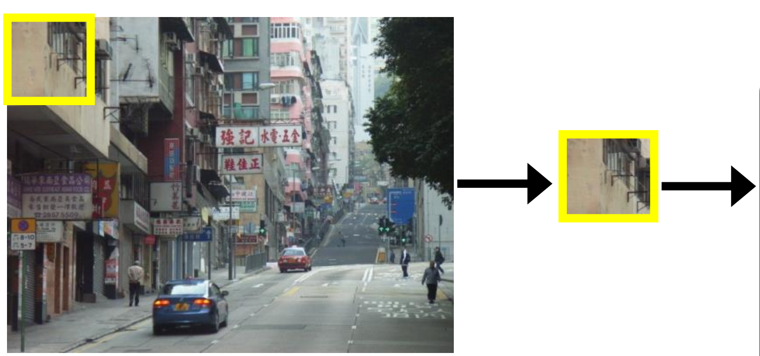


Window-template-based object detection: recap

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

Given new image:
1. Slide window
2. Score by classifier





Kristen Grauman

Dalal-Triggs pedestrian detector

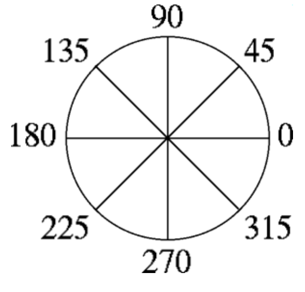
1. Extract fixed-sized (64x128 pixel) window at multiple positions and scales
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

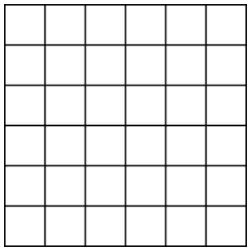
Histograms of oriented gradients (HOG)

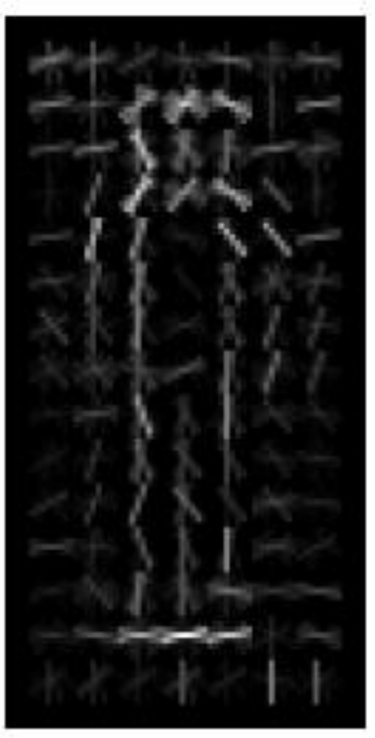
Divide image into 8x8 regions

Orientation: 9 bins
(for unsigned angles)



Histograms in
8x8 pixel cells



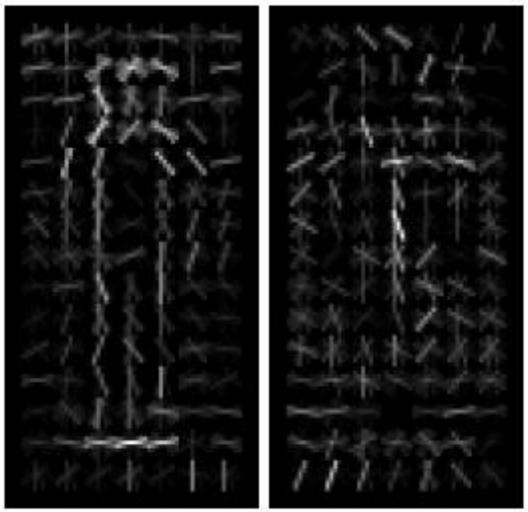


Votes weighted by magnitude

Adapted from Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Train SVM for pedestrian detection using HoG

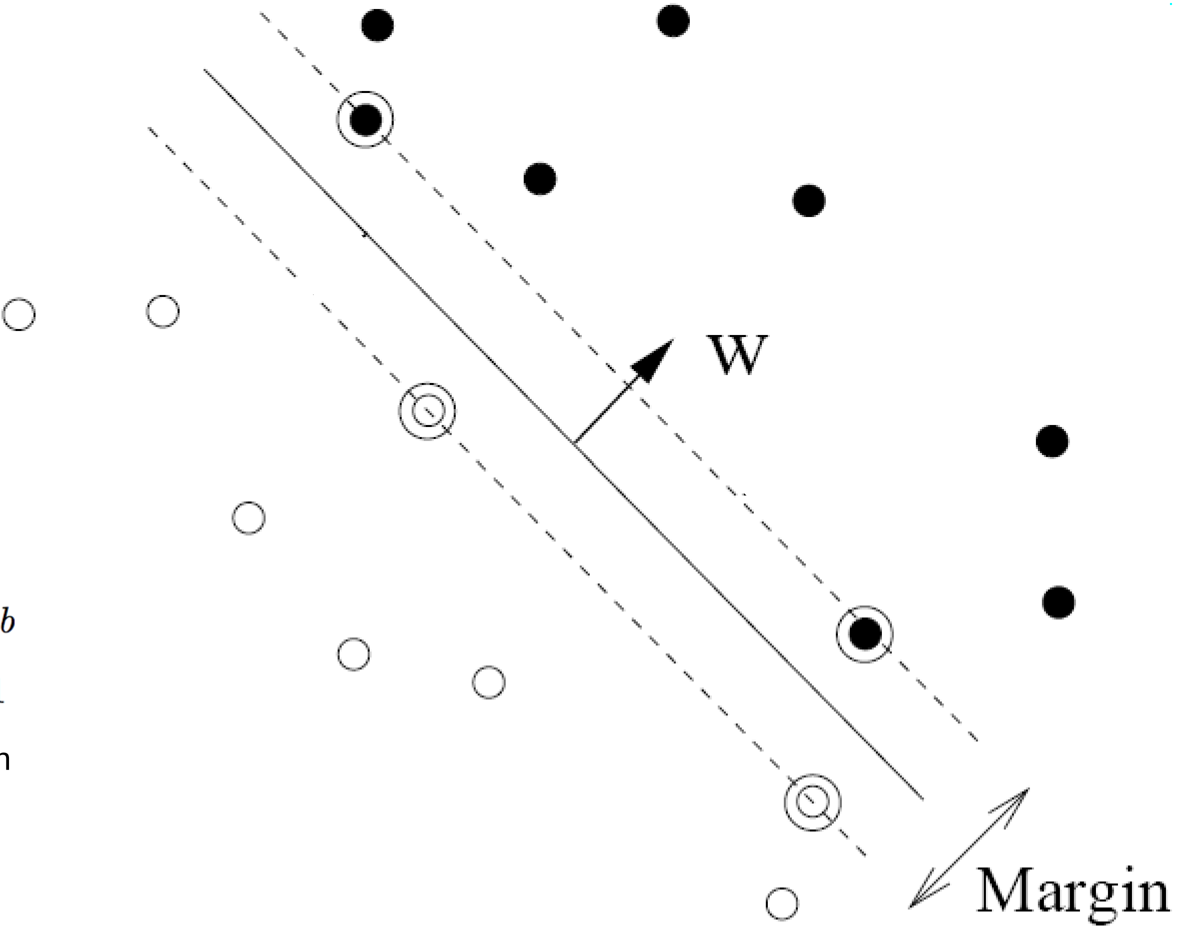


pos w neg w

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

$$sign(0.16) = 1$$

=> pedestrian

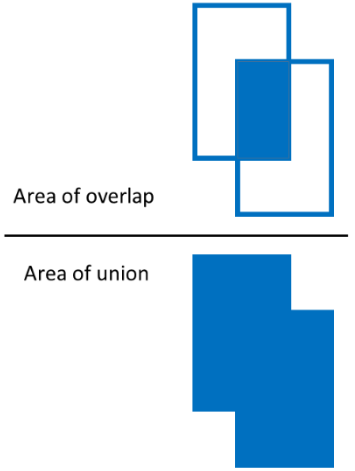


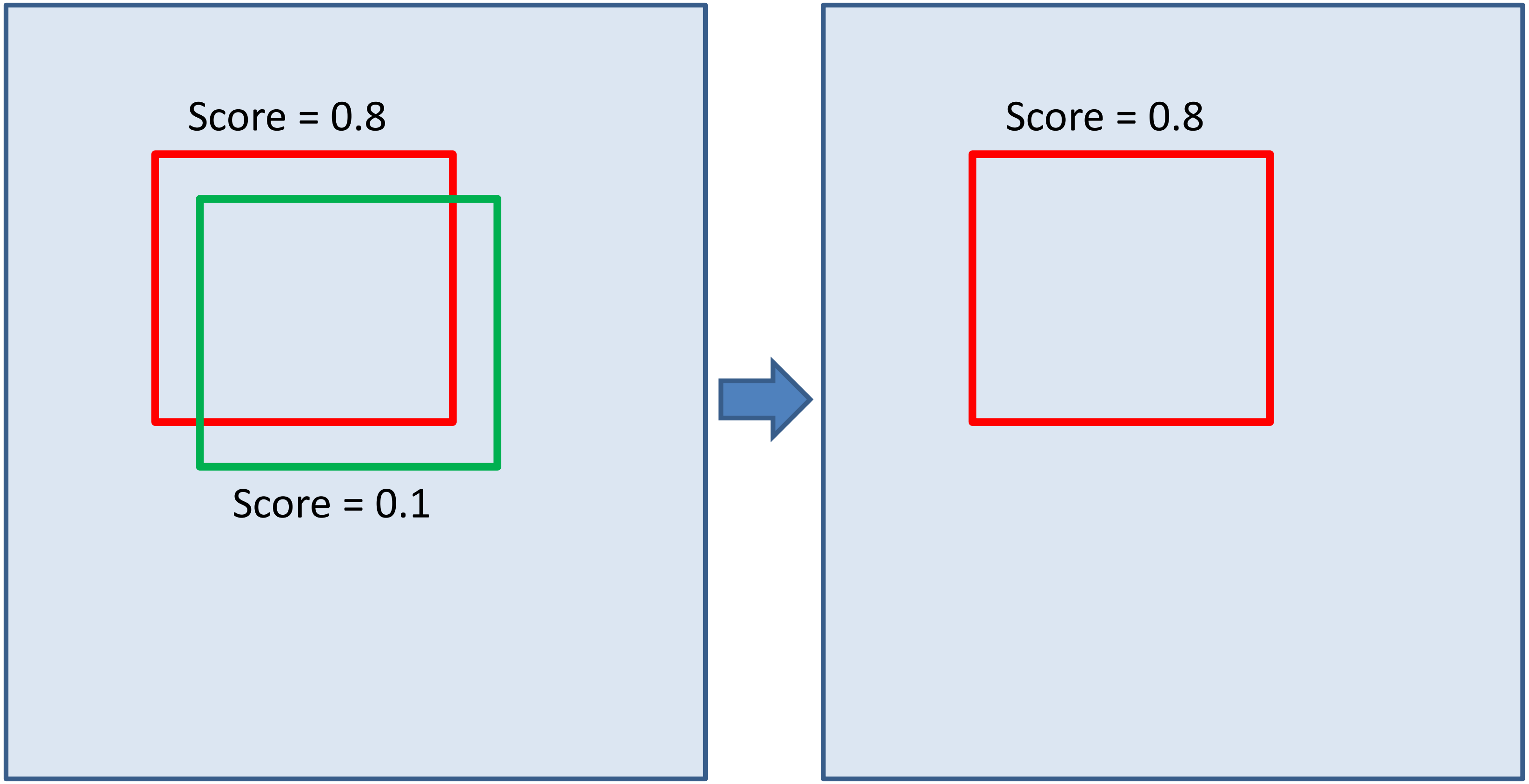
Adapted from Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

Remove overlapping detections

Non-max suppression





Adapted from Derek Hoiem

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

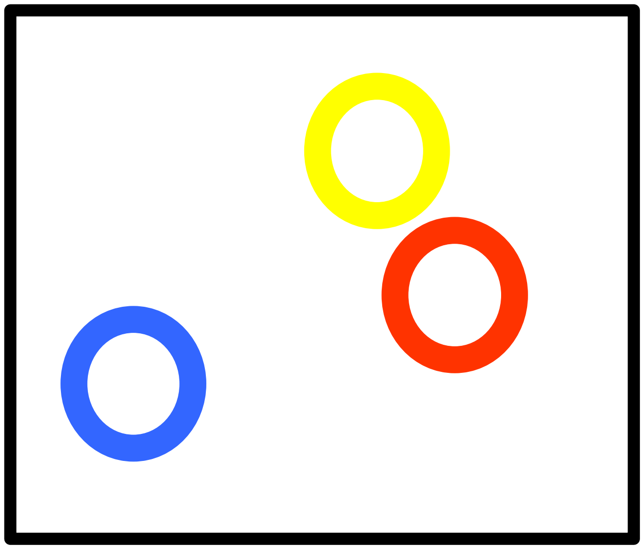
Define object by collection of parts modeled by

1. Appearance
2. Spatial configuration

Slide credit: Rob Fergus

How to model spatial relations?

- One extreme: fixed template



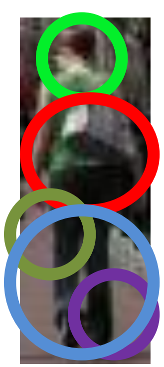
Derek Hoiem

Fixed part-based template

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

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

Non-object



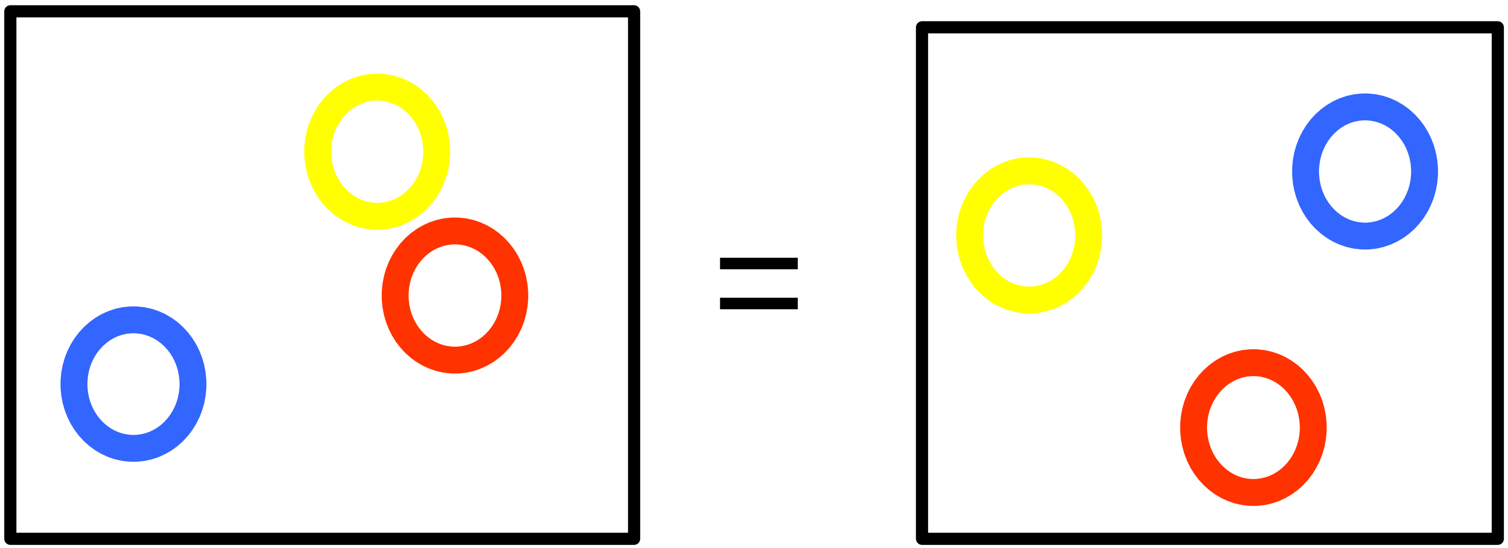
$+4 +1 +0.5 +3 +0.5 = 10.5 \overset{?}{>} 7.5$

Object

Derek Hoiem

How to model spatial relations?

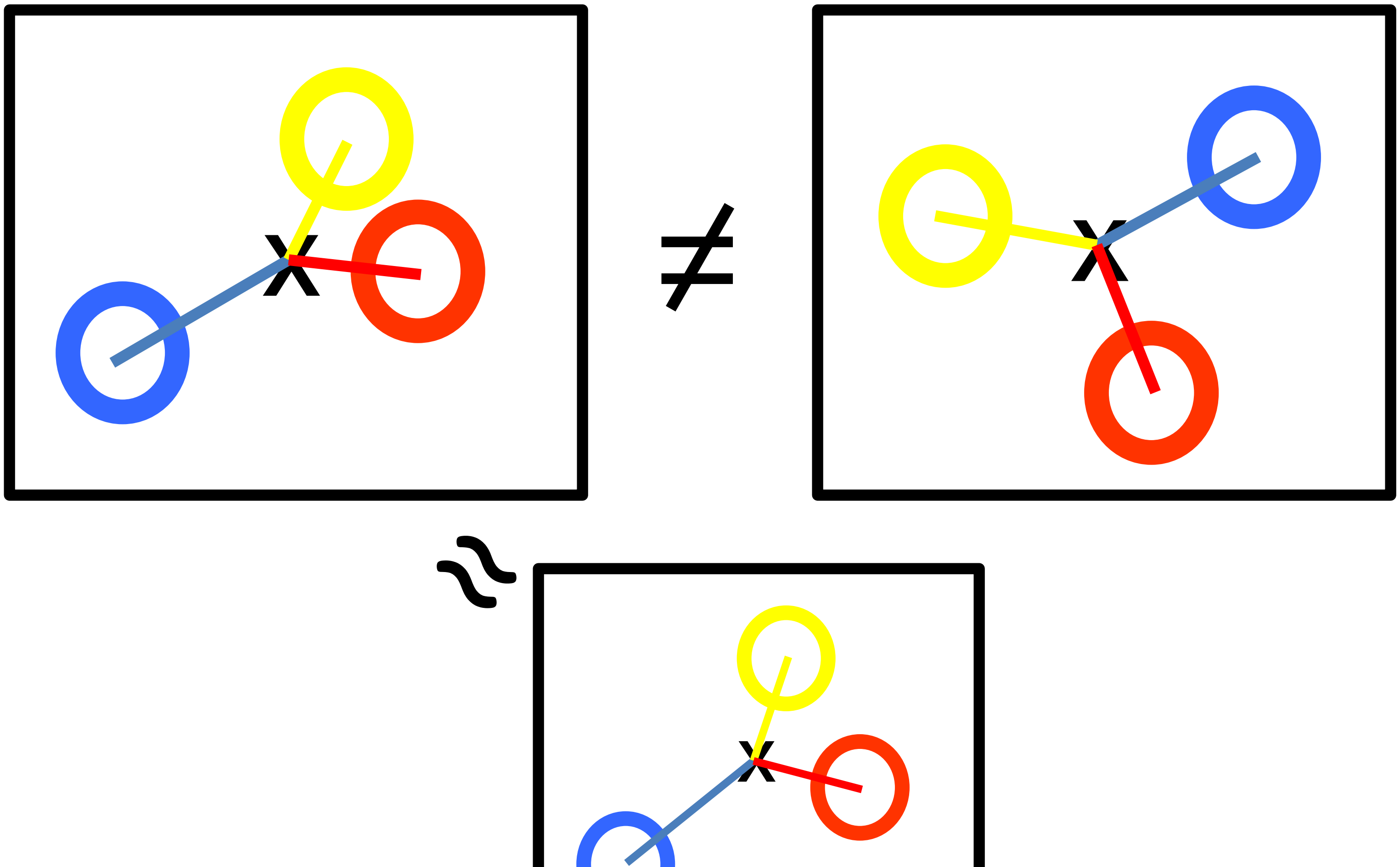
- Another extreme: bag of words



Derek Hoiem

How to model spatial relations?

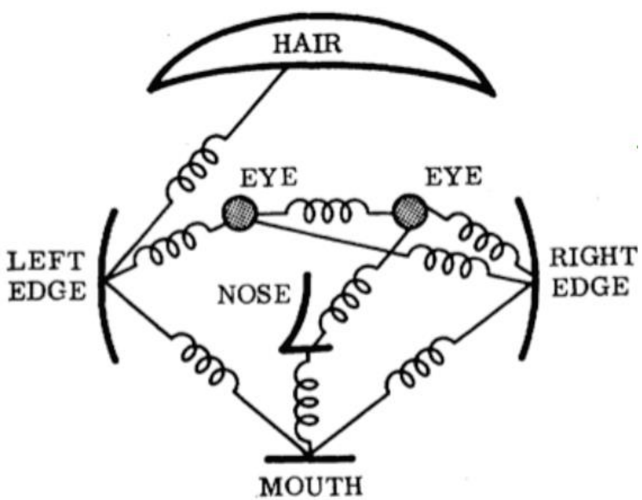
- Star-shaped model

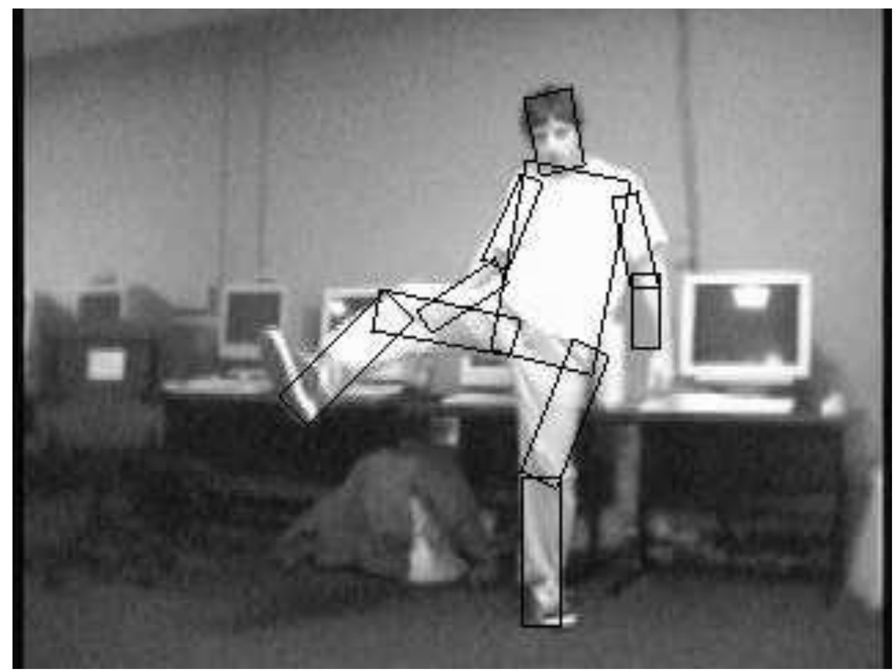


Derek Hoiem

Parts-based Models

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

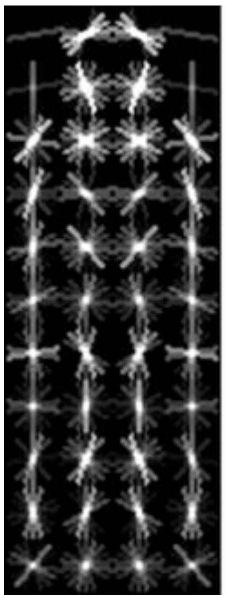




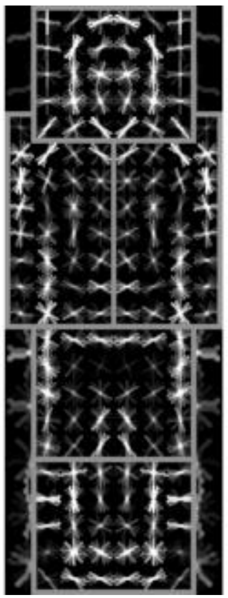
Adapted from Derek Hoiem, images from Felzenszwalb

Discriminative part-based models

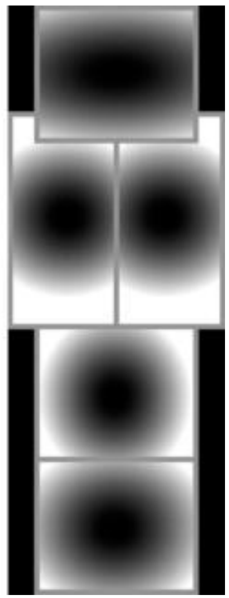
Root filter

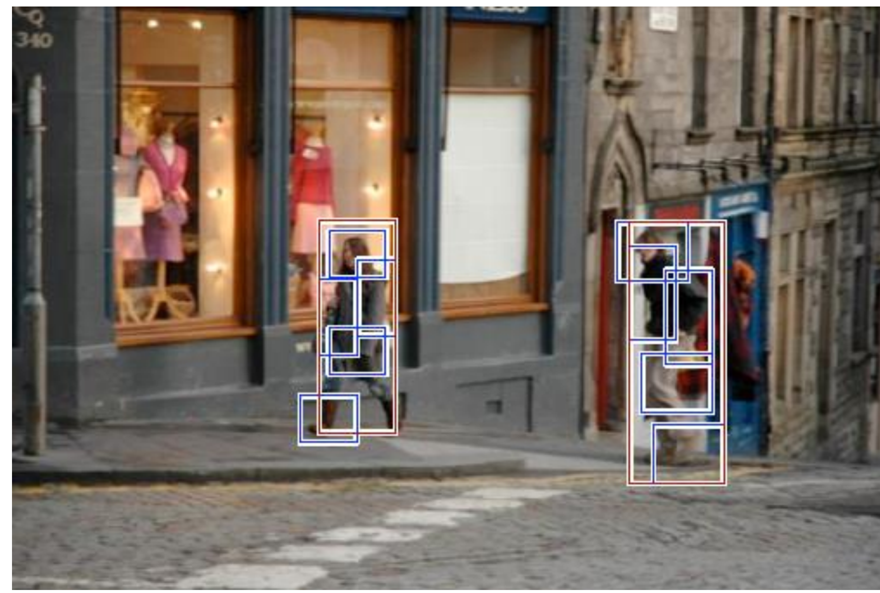


Part filters



Deformation weights



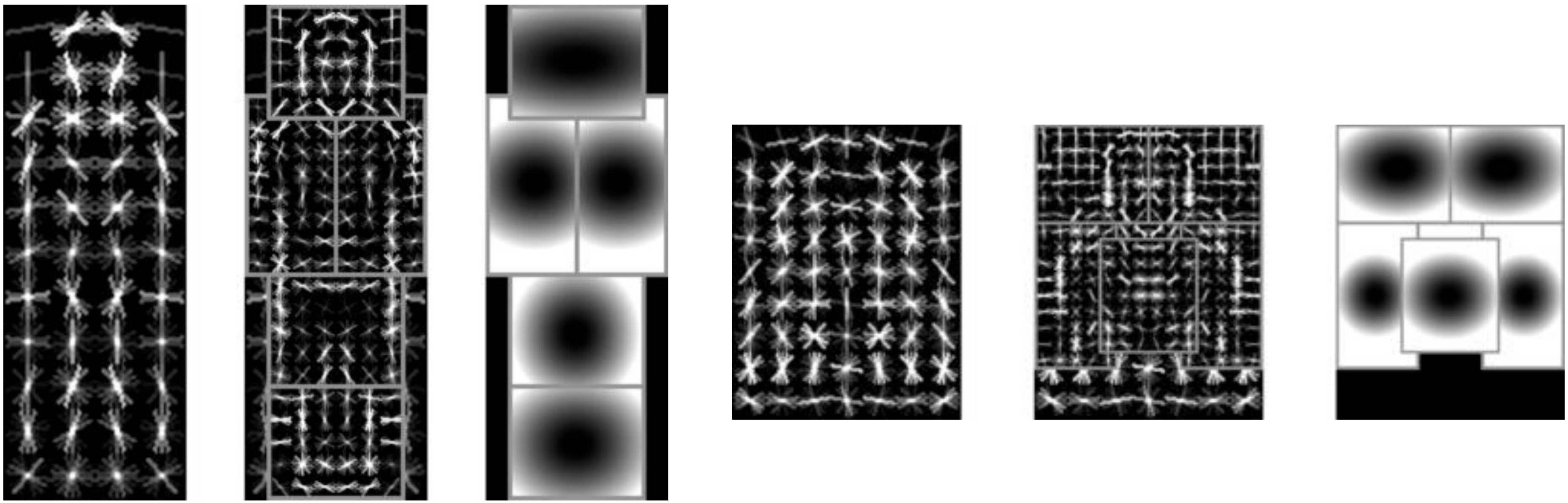


P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, [Object Detection with Discriminatively Trained Part Based Models](), PAMI 32(9), 2010

Lana Lazebnik

Discriminative part-based models

Multiple components

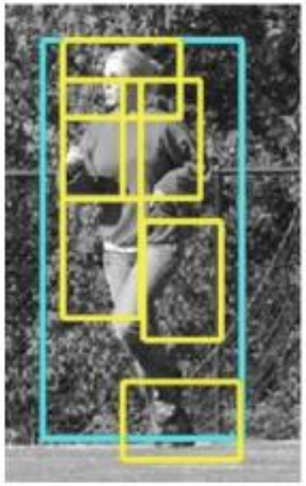


P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, [Object Detection with Discriminatively Trained Part Based Models](), PAMI 32(9), 2010

Lana Lazebnik

Scoring an object hypothesis

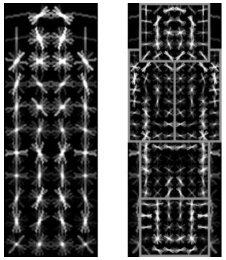
- The score of a hypothesis is the sum of appearance scores minus the sum of deformation costs

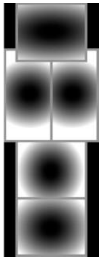


$$z = (p_0, \ldots, p_n)$$

p_0 : location of root

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





part loc (where we see part)

anchor loc (where we *expect* to see part)

$$(dx_i, dy_i) = \overbrace{(x_i, y_i)} - \overbrace{(2(x_0, y_0) + v_i)}$$

Displacements

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

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

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

Appearance weights

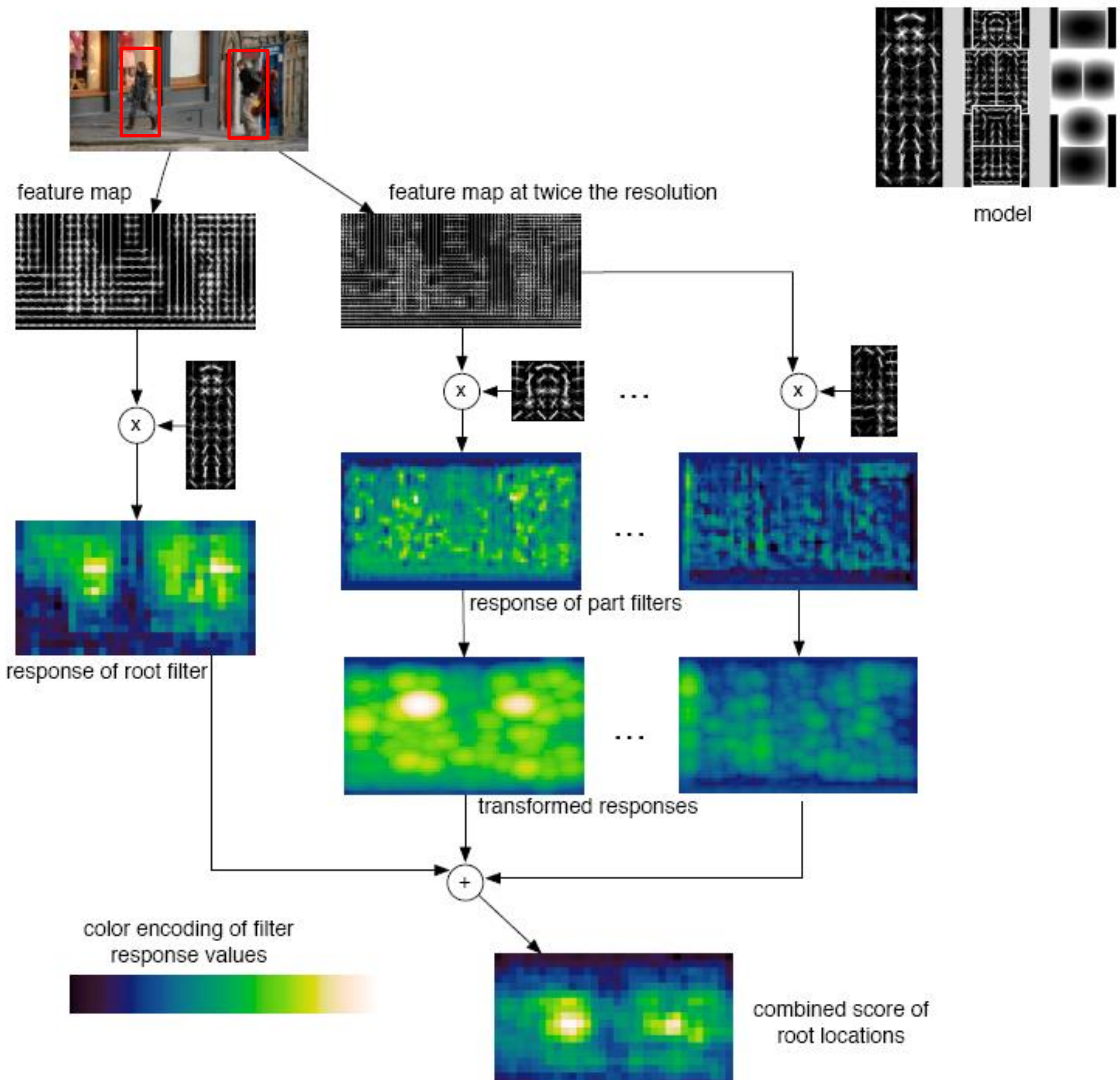
Part features

Deformation weights

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

Felzenszwalb et al.

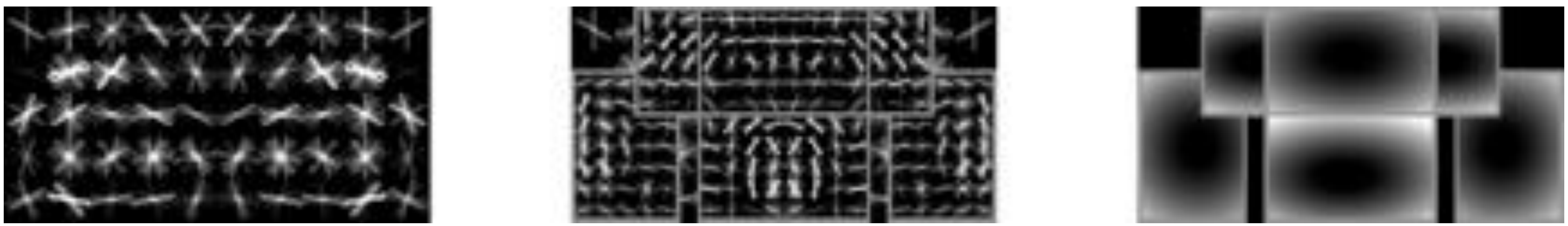
Detection



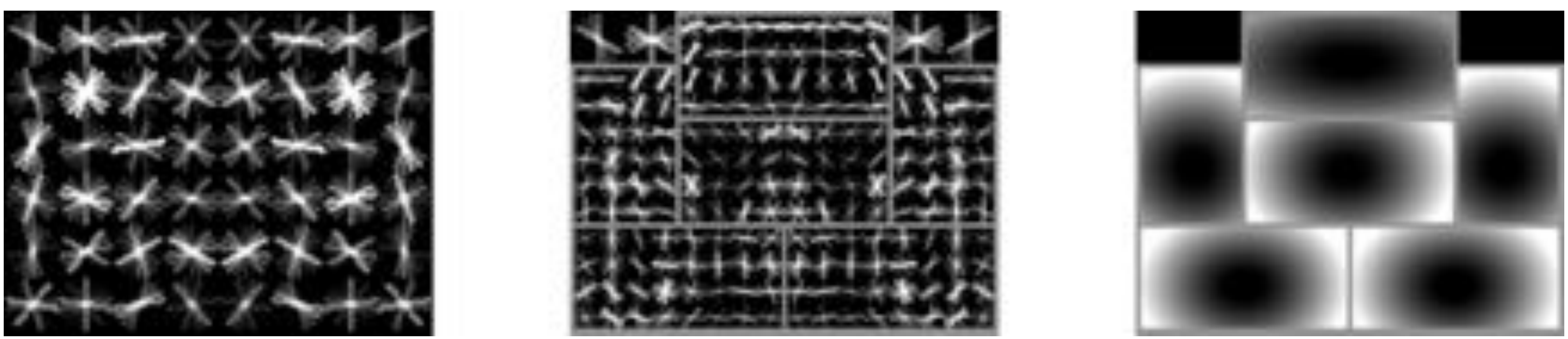
Felzenszwalb et al.

Car model

Component 1



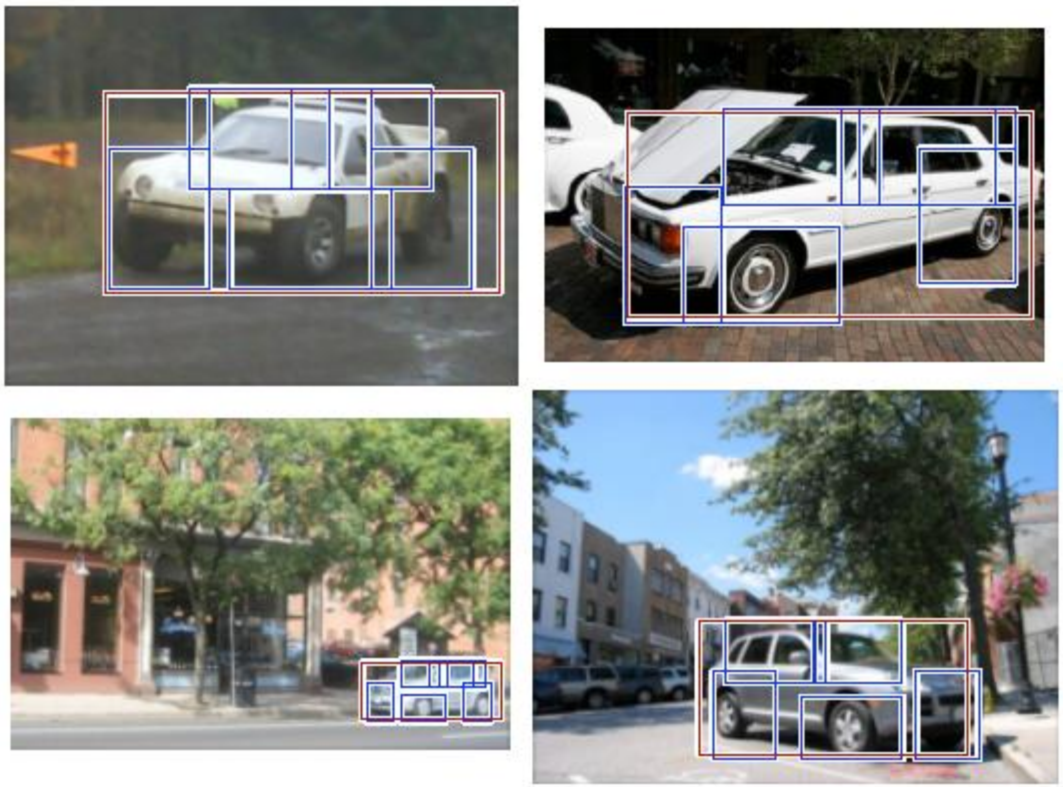
Component 2



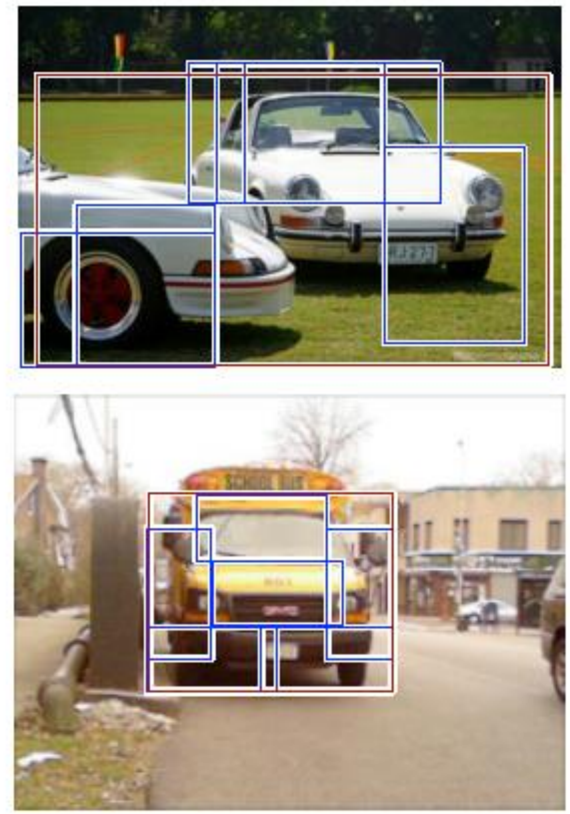
Lana Lazebnik

Car detections

high scoring true positives

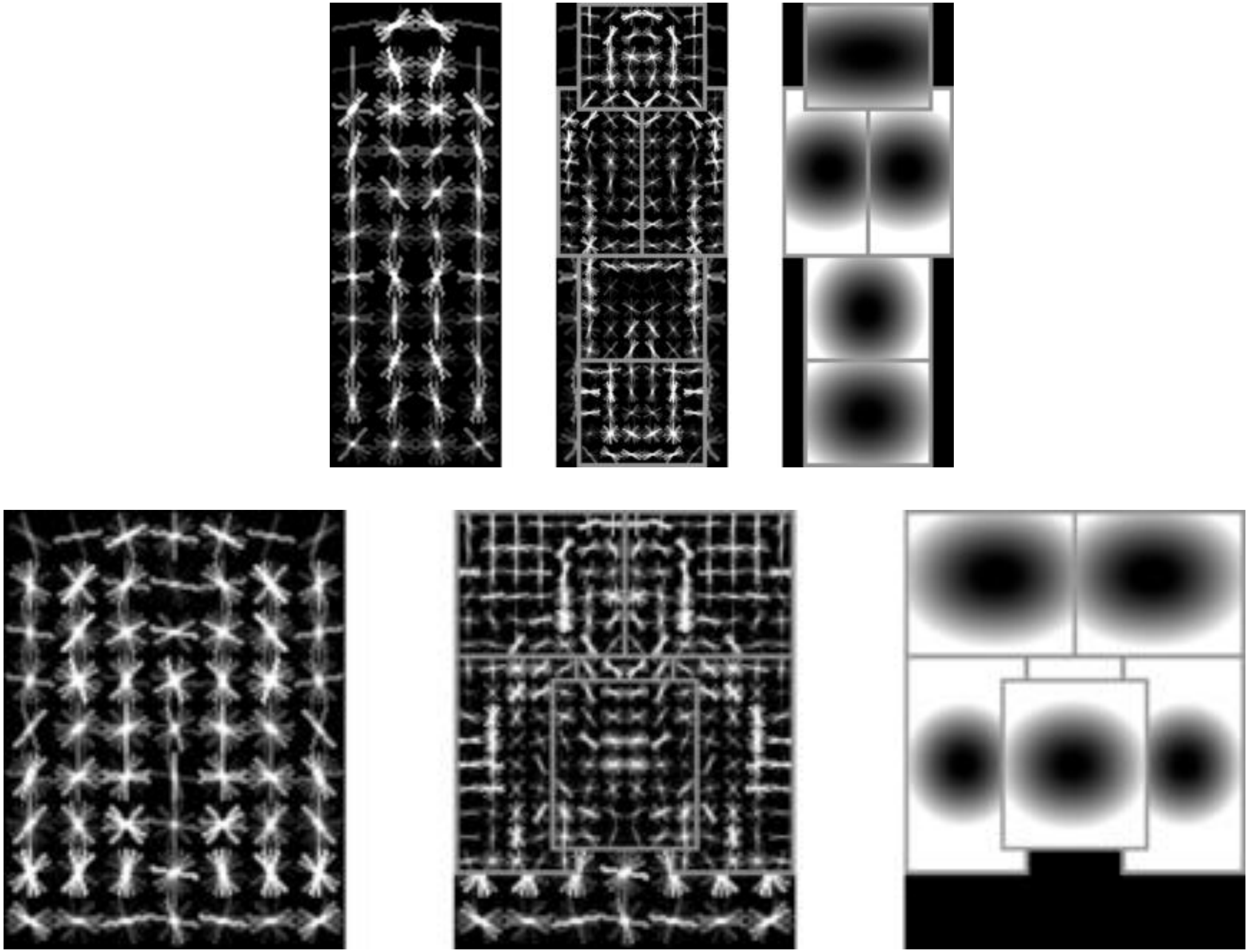


high scoring false positives



Lana Lazebnik

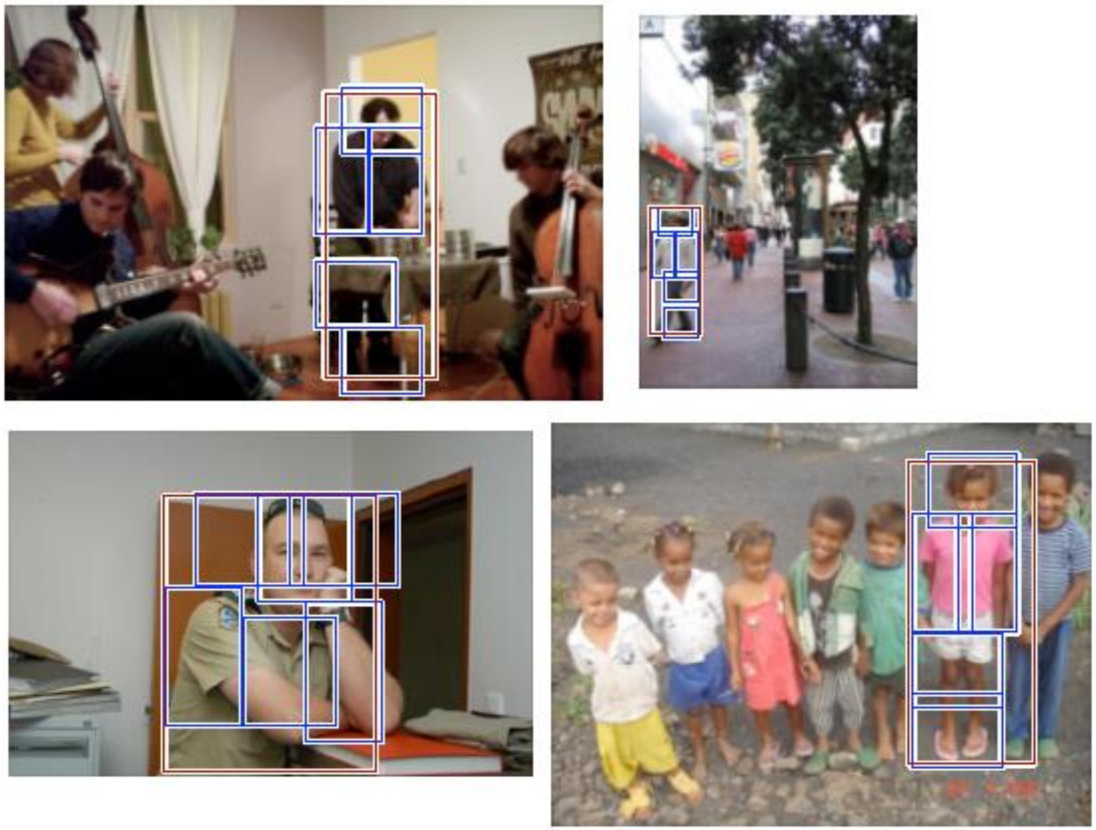
Person model



Lana Lazebnik

Person detections

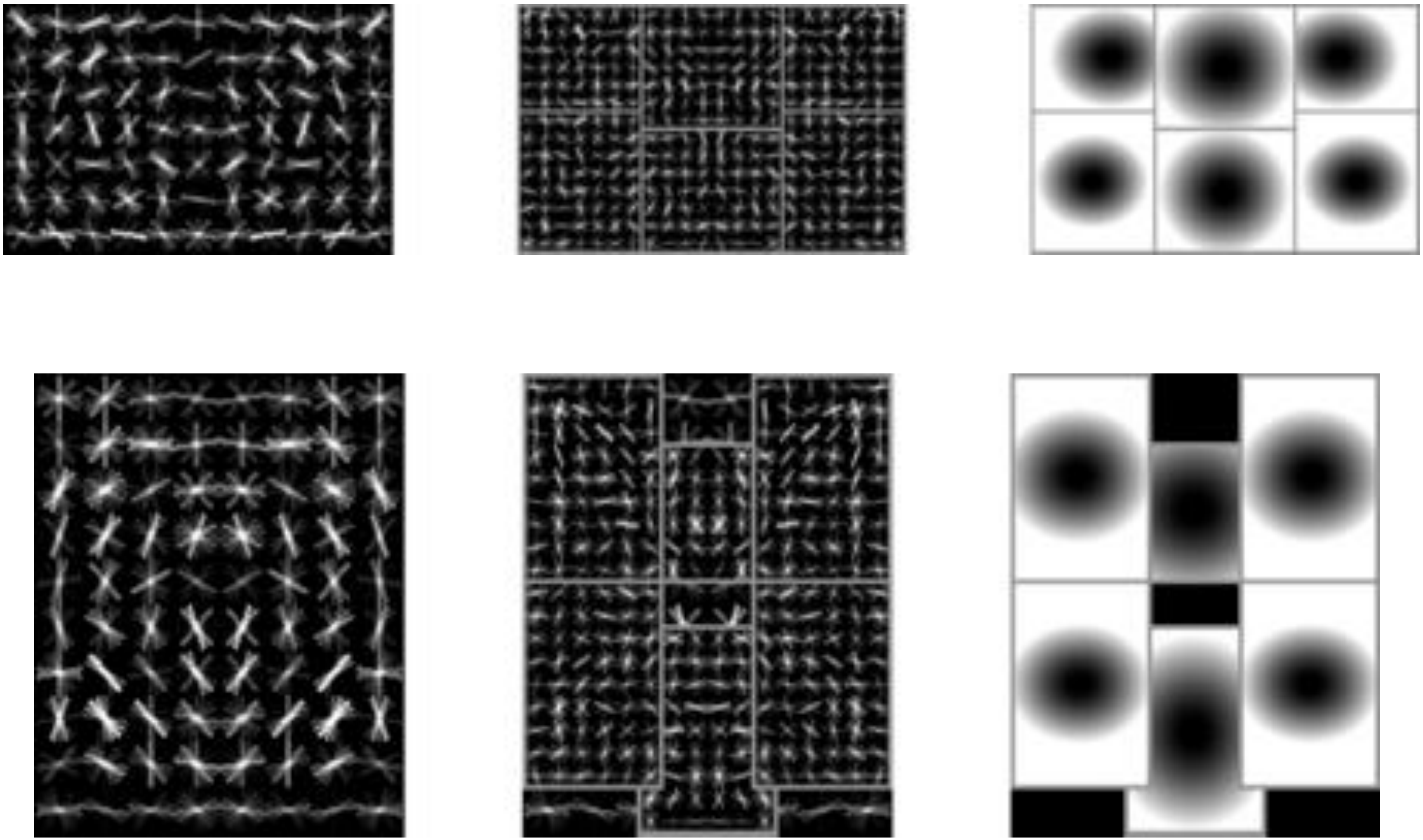
high scoring true positives



high scoring false positives
(not enough overlap)

Lana Lazebnik

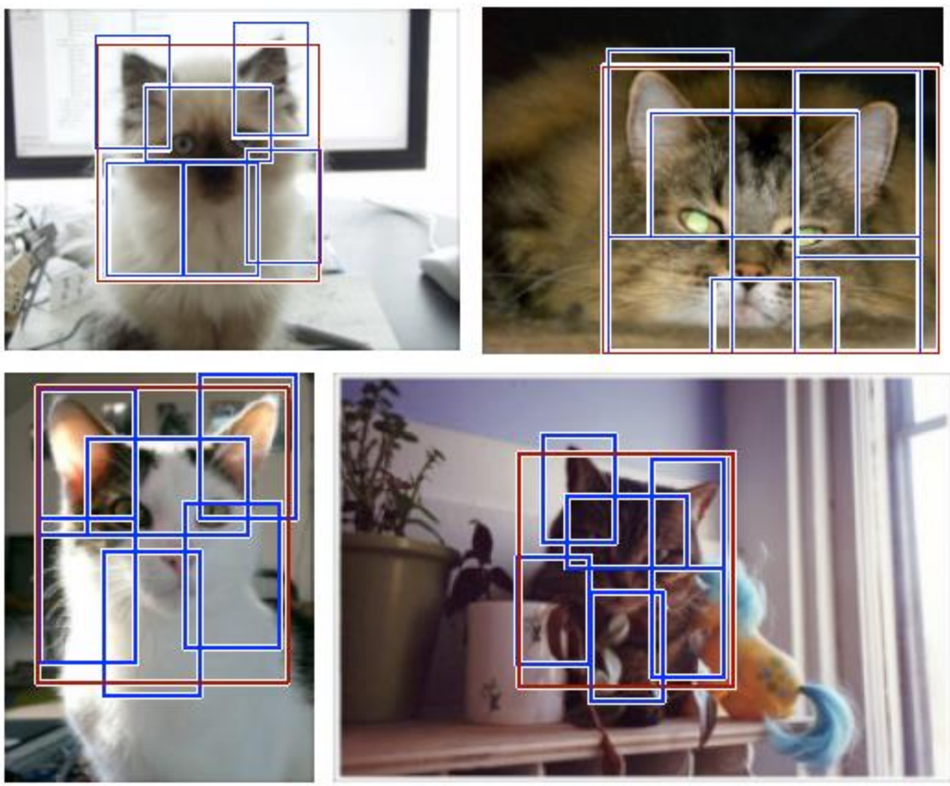
Cat model



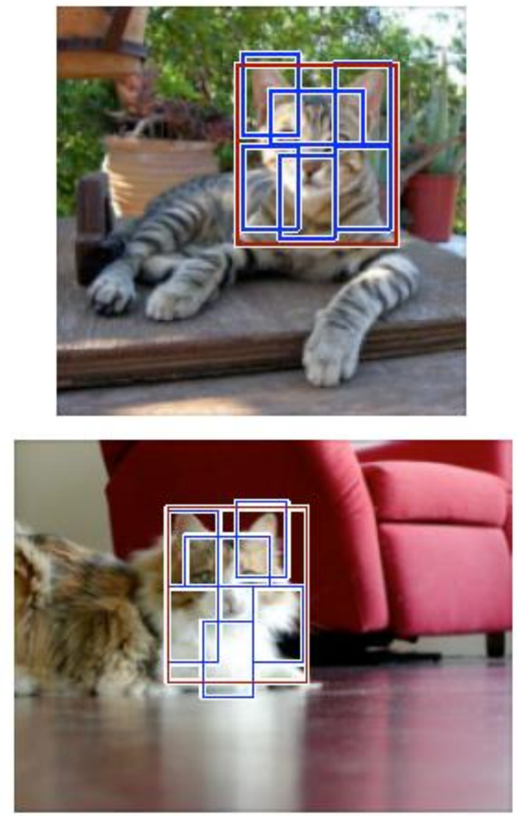
Lana Lazebnik

Cat detections

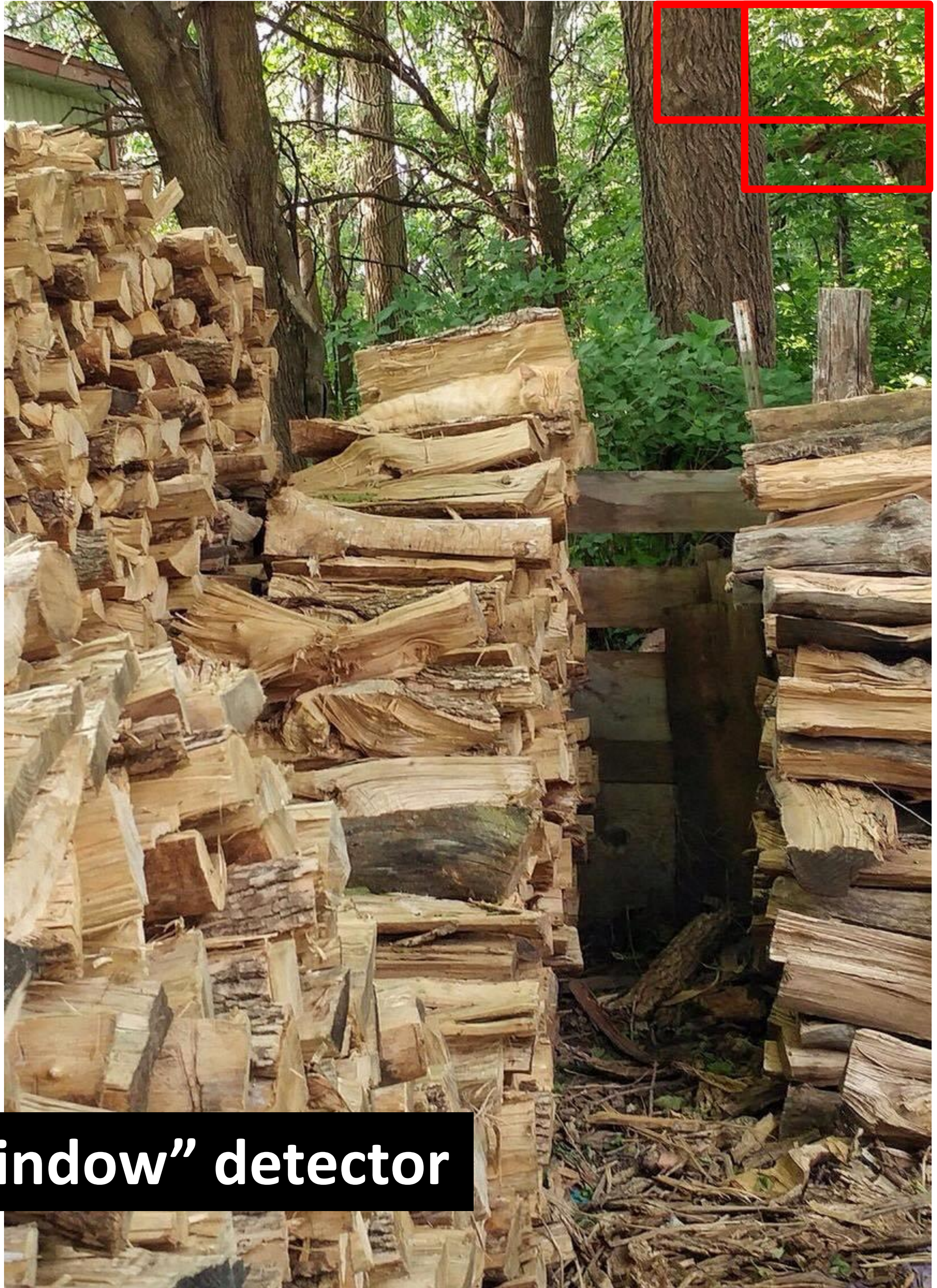
high scoring true positives



high scoring false positives
(not enough overlap)



Lana Lazebnik



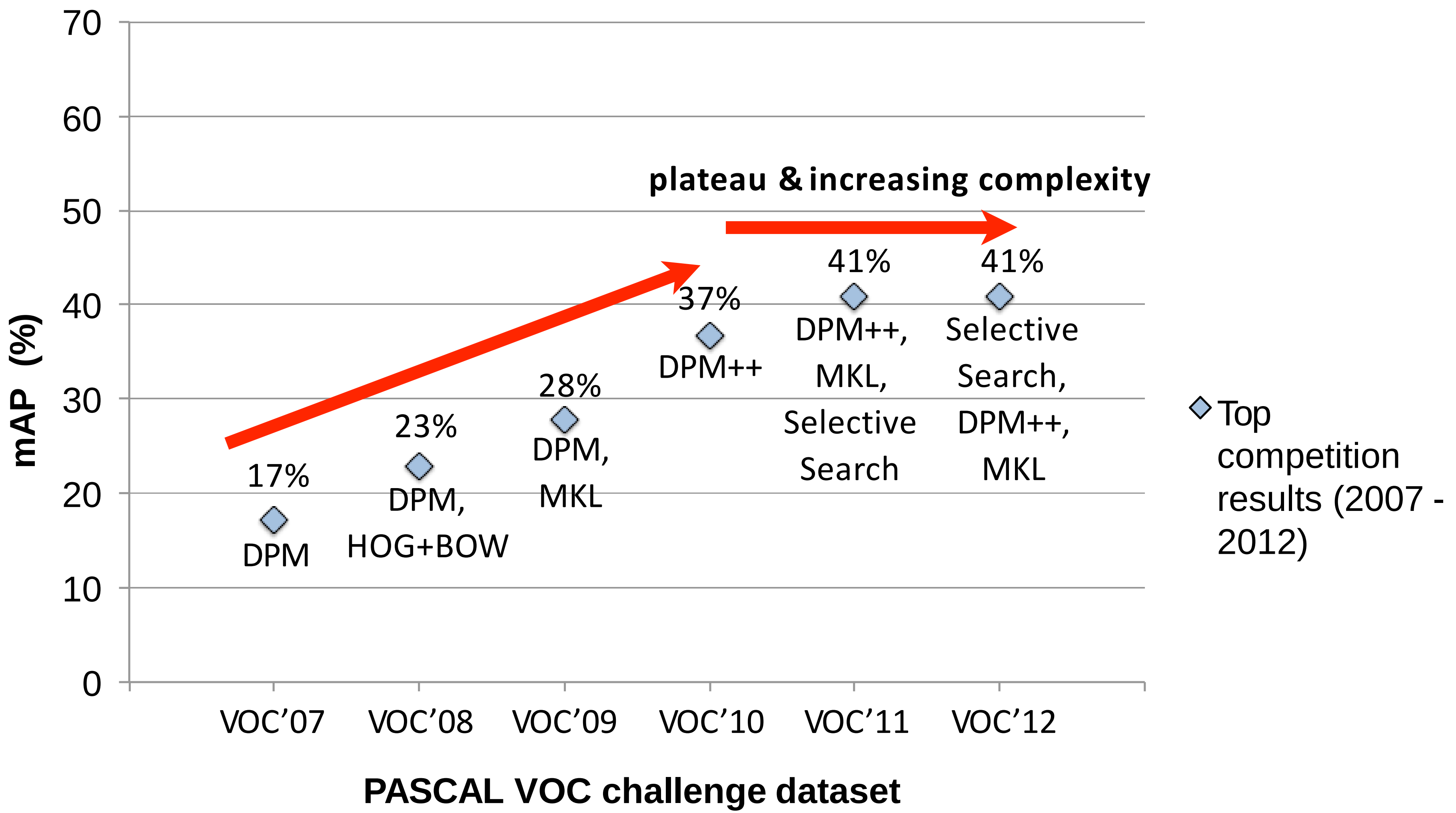
"Sliding window" detector

Plan for this lecture

- Fully supervised detection
  - Pre-CNN: Deformable part models
  - Detection with region proposals: R-CNN, Fast/er R-CNN
  - Detection without region proposals: YOLO
  - Semantic and instance segmentation: FCN, Mask R-CNN
- Weak or out-of-domain supervision
  - Weakly supervised object detection
  - Domain adaptation

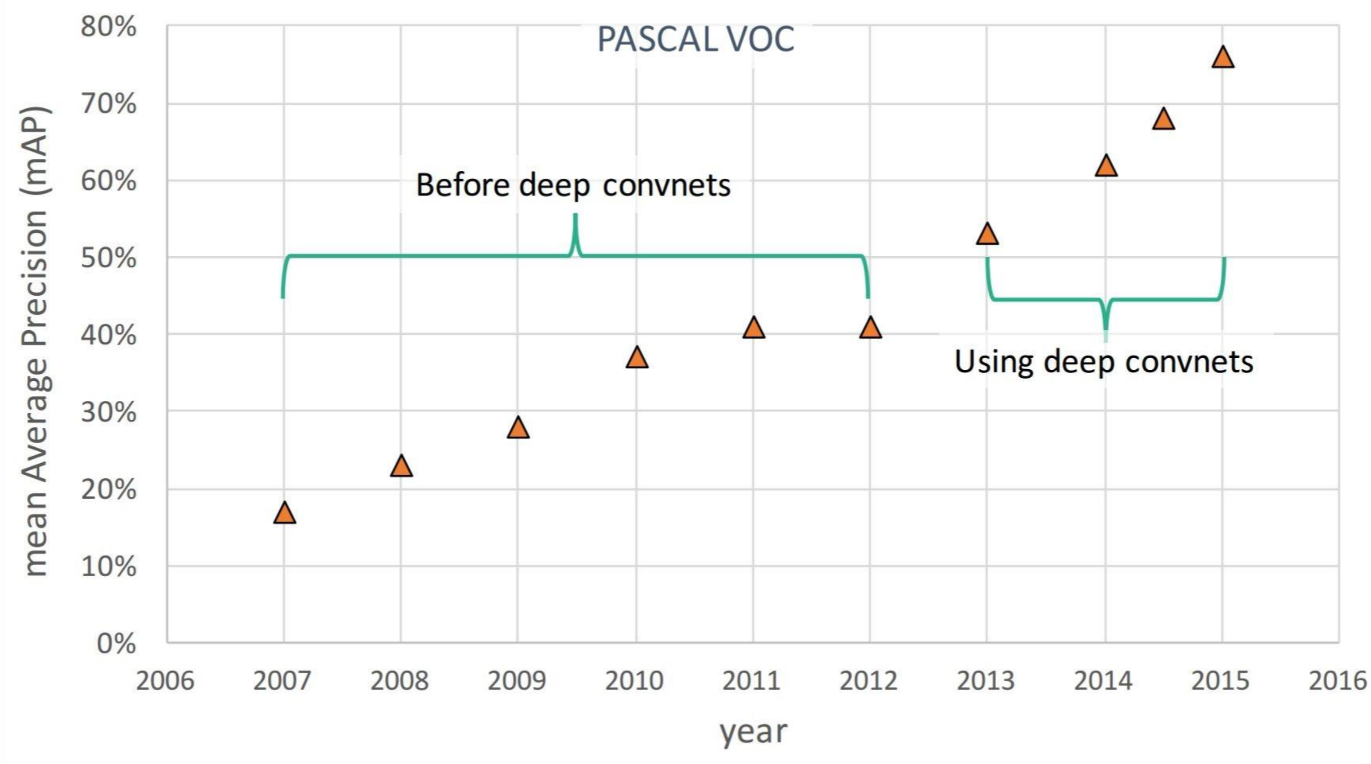
Complexity and the plateau

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



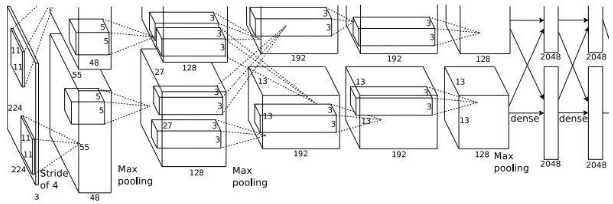
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

Impact of Deep Learning

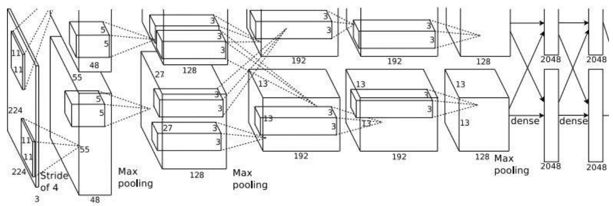


Slide by: Justin Johnson

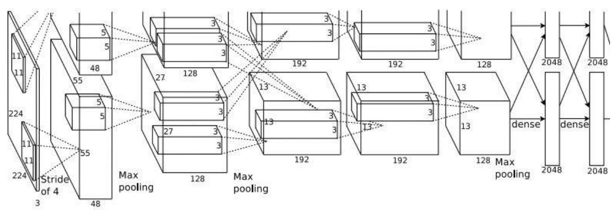
Object Detection as Regression?



CAT: (x, y, w, h)



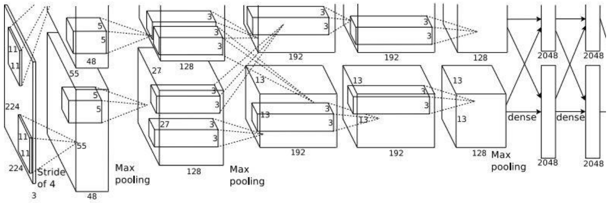
DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

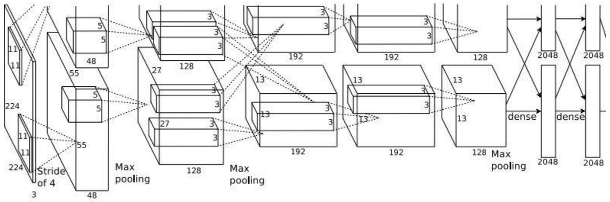
Slide by: Justin Johnson

Object Detection as Regression?



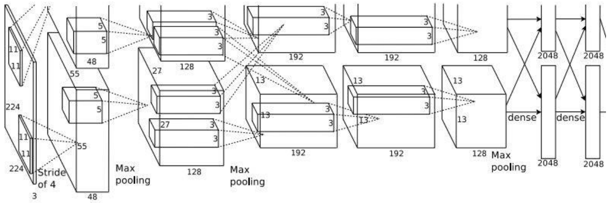
CAT: (x, y, w, h)

4 numbers



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

16 numbers



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

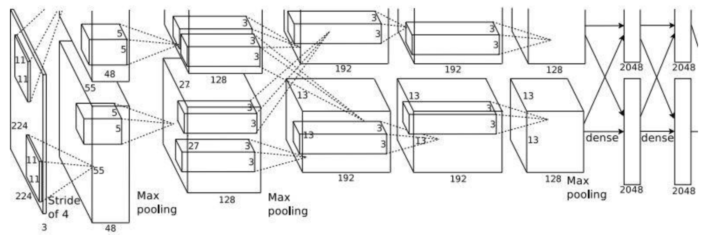
Many numbers!

Each image needs a different number of outputs!

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

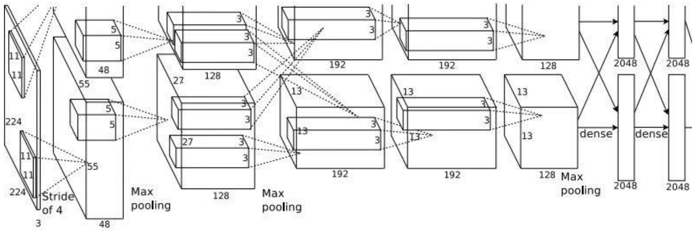
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



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

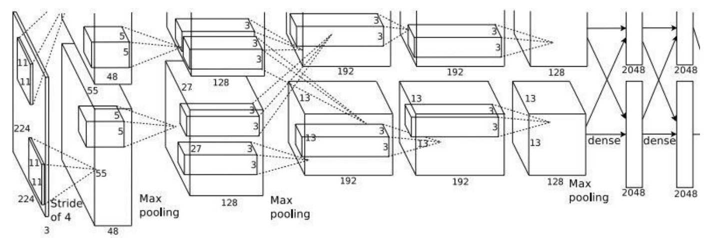


Dog? YES
Cat? NO
Background? NO

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

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

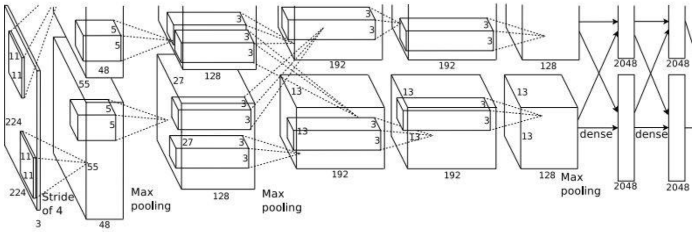


Dog? YES
Cat? NO
Background? NO

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

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

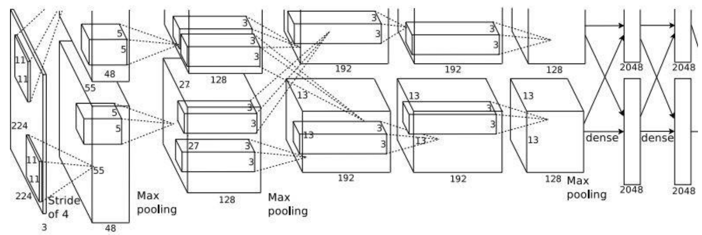


Dog? NO
Cat? YES
Background? NO

Slide by: Justin Johnson

Object Detection as Classification: Sliding Window

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



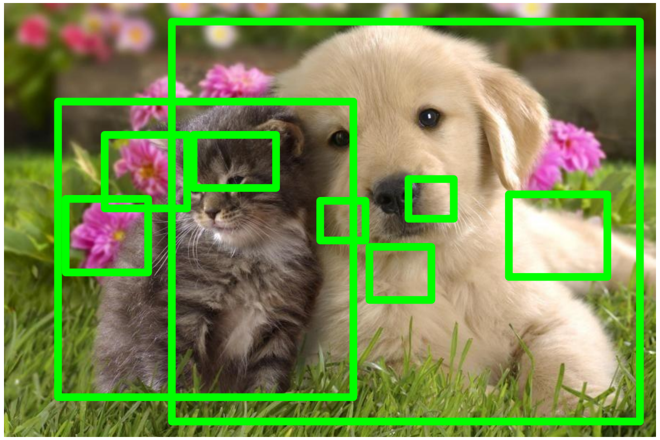
Dog? NO
Cat? YES
Background? NO

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

Slide by: Justin Johnson

Region Proposals

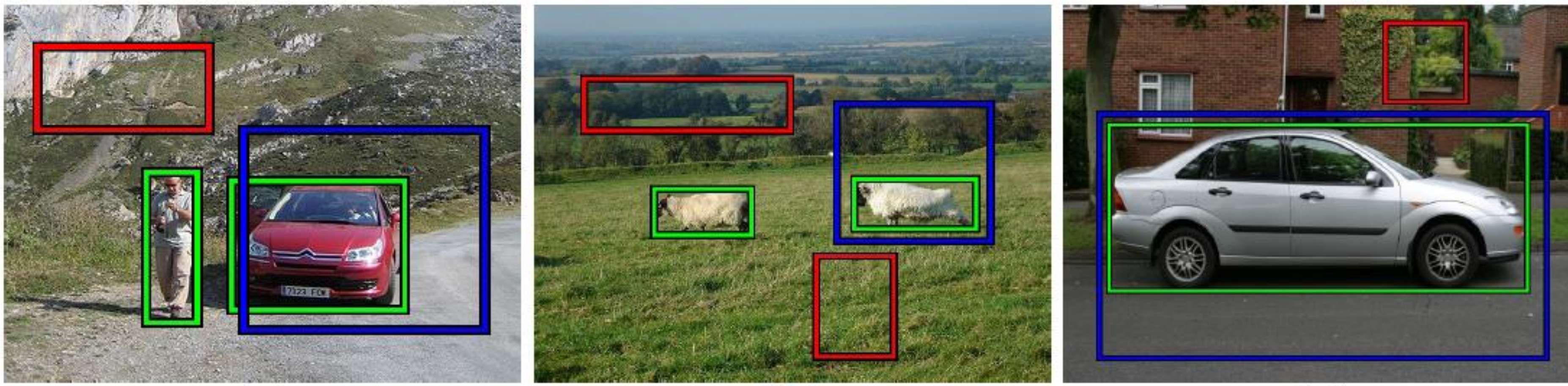
- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run; e.g. Selective Search gives 1000 region proposals in a few seconds on CPU



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

Slide by: Justin Johnson

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.*

Alexe et al., "Measuring the objectness of image windows", PAMI 2012 and CVPR 2010

Proposals cue: color contrast at boundary

(a) (b) (c)

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.*

Alexe et al., “Measuring the objectness of image windows”, PAMI 2012 and CVPR 2010

Proposals cue: no segments “straddling” the object box

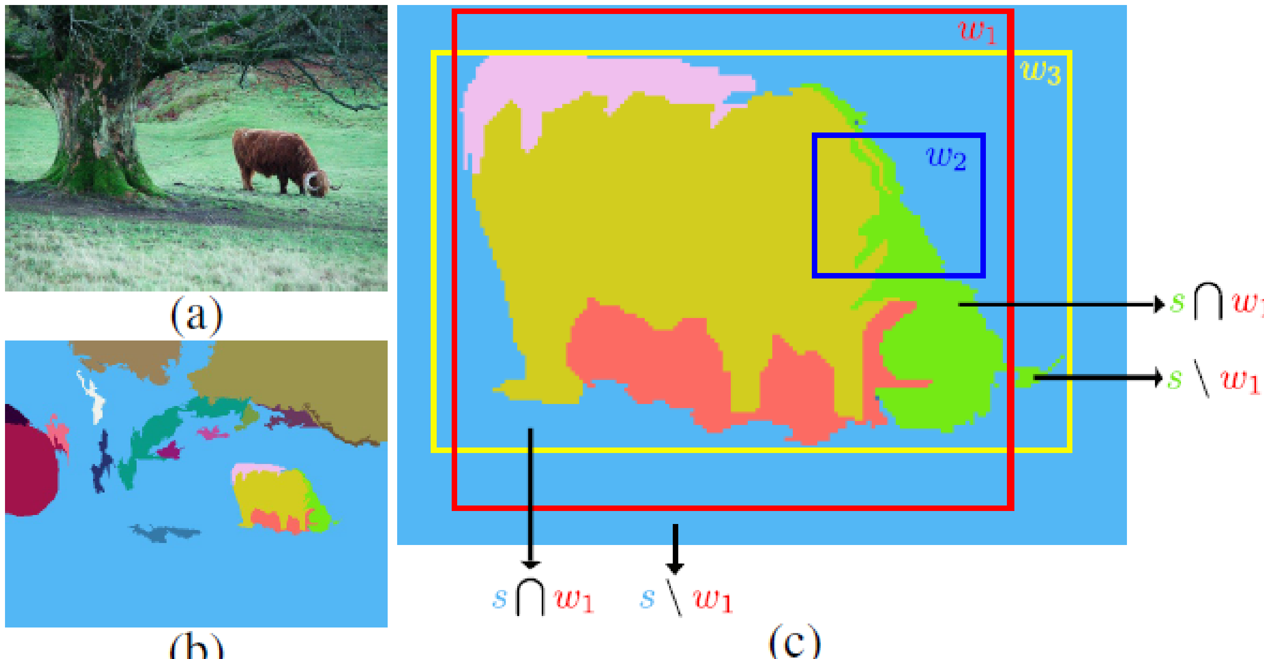
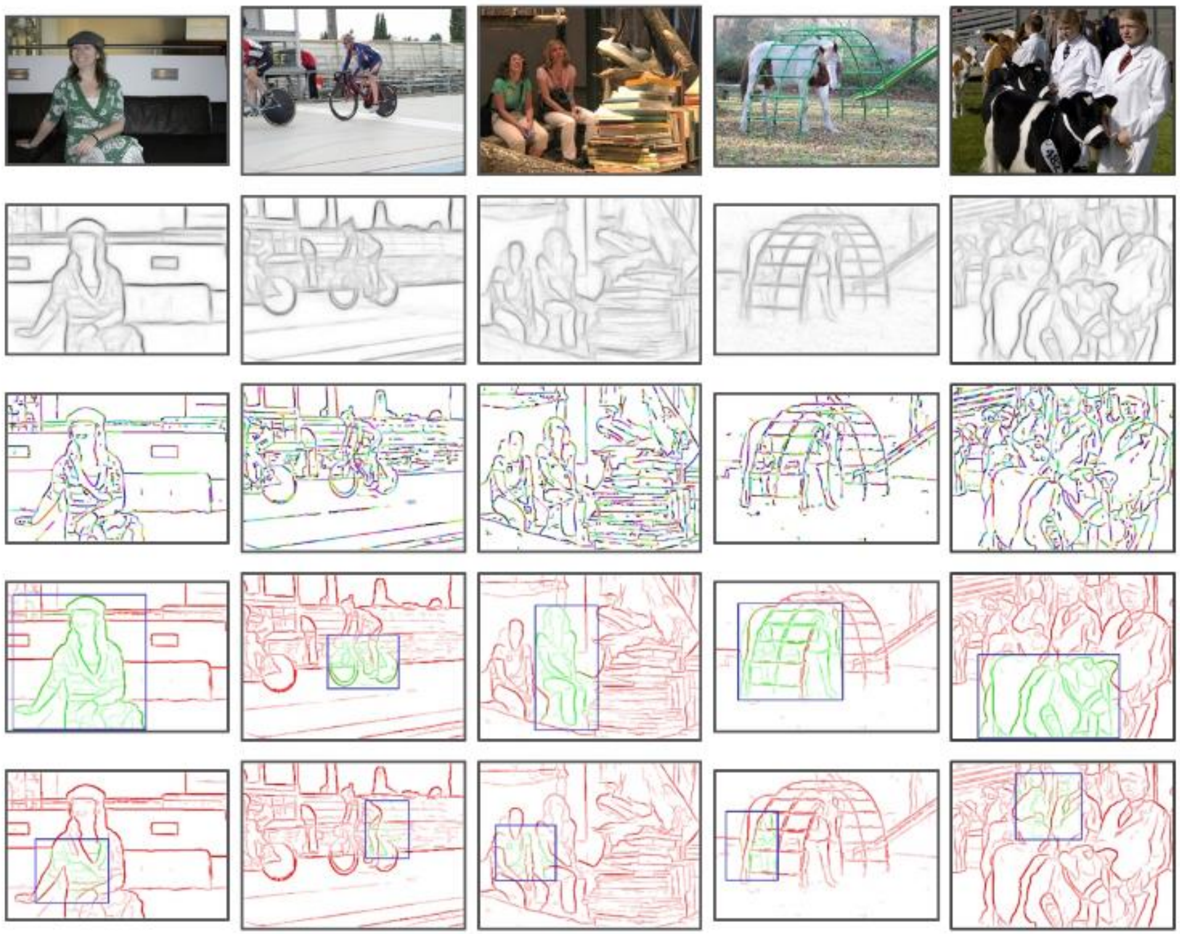


Fig. 5: **The SS cue.** *Given the segmentation (b) of image (a), for a window w we compute* $\mathrm{SS}(\mathrm{w}, \theta_{\mathrm{SS}})$ *(eq. 4). In (c), most of the surface of w_1 is covered by superpixels contained almost entirely inside it. Instead, all superpixels passing by w_2 continue largely 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.*

Alexe et al., “Measuring the objectness of image windows”, PAMI 2012 and CVPR 2010

Proposals cue: many edges wholly contained inside box



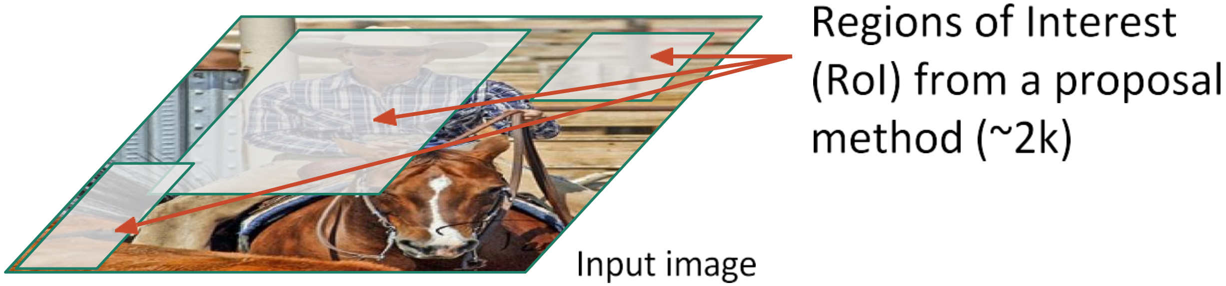
Zitnick and Dollar, "Edge Boxes: Locating Object Proposals from Edges", ECCV 2014

R-CNN

Input image

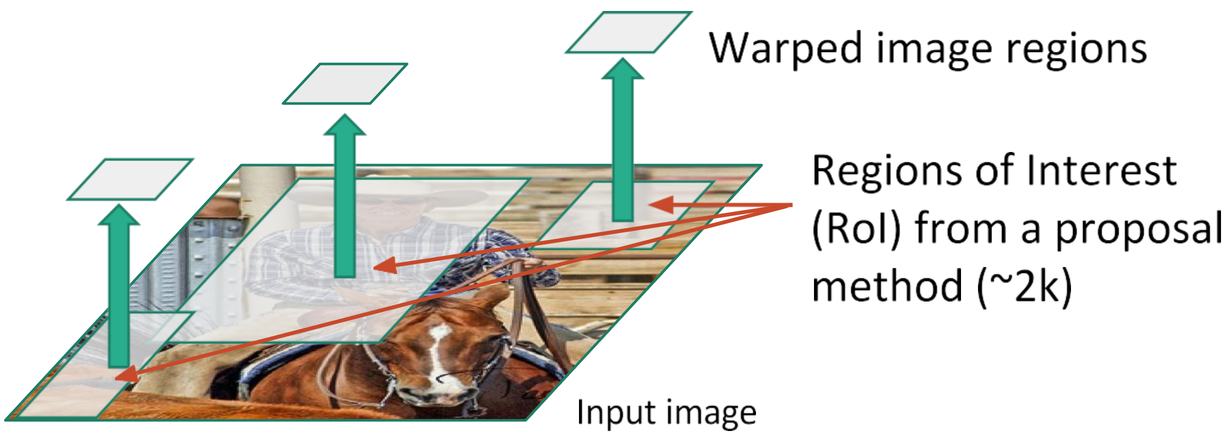
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



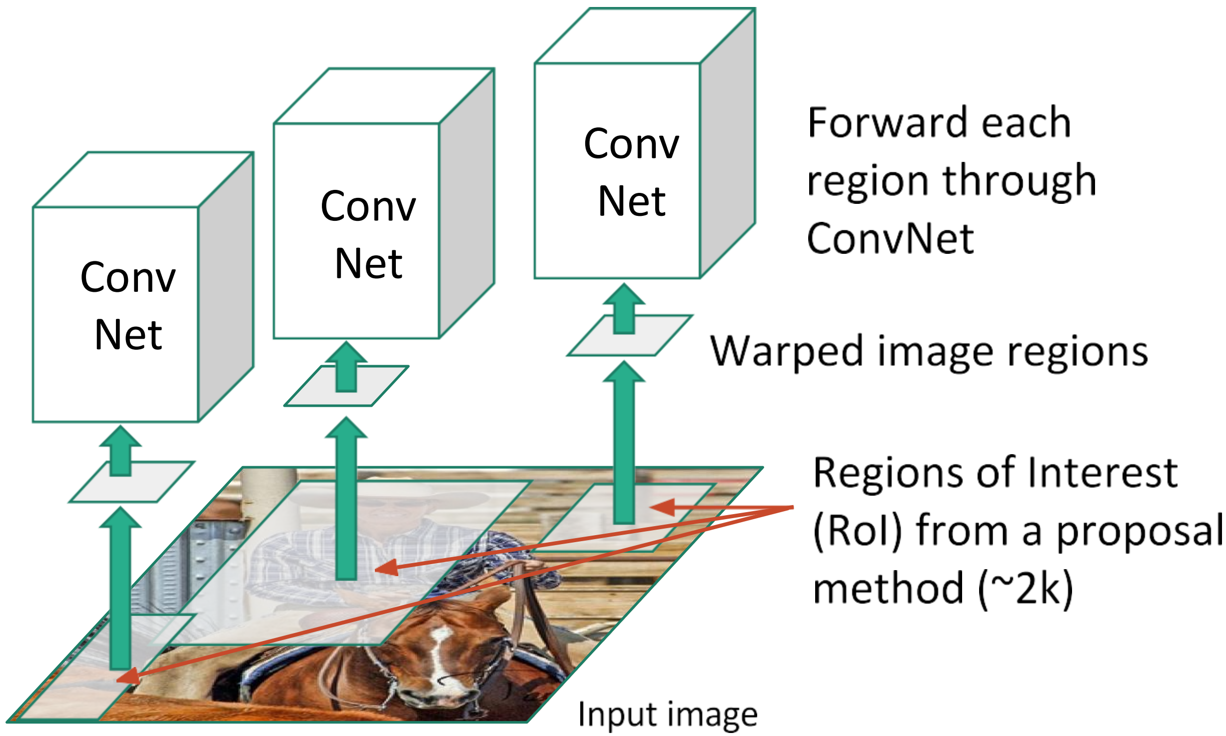
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



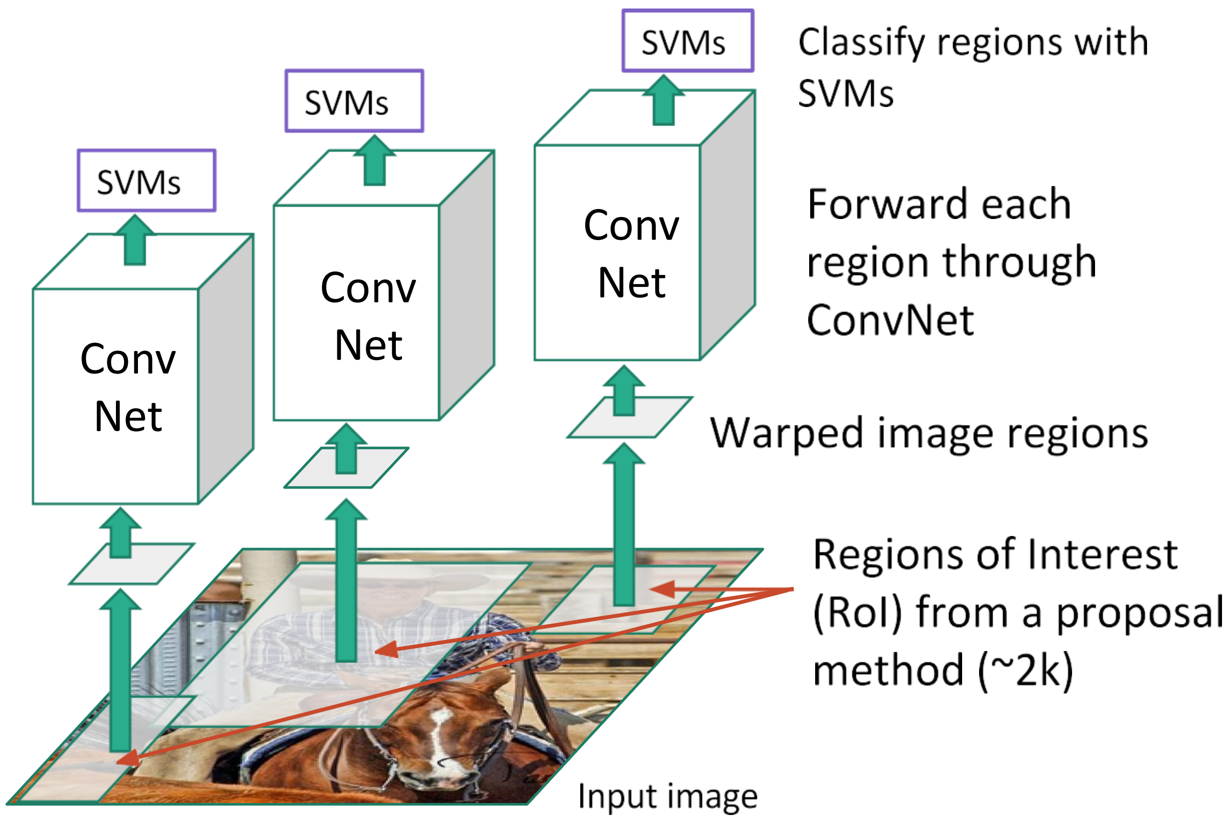
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



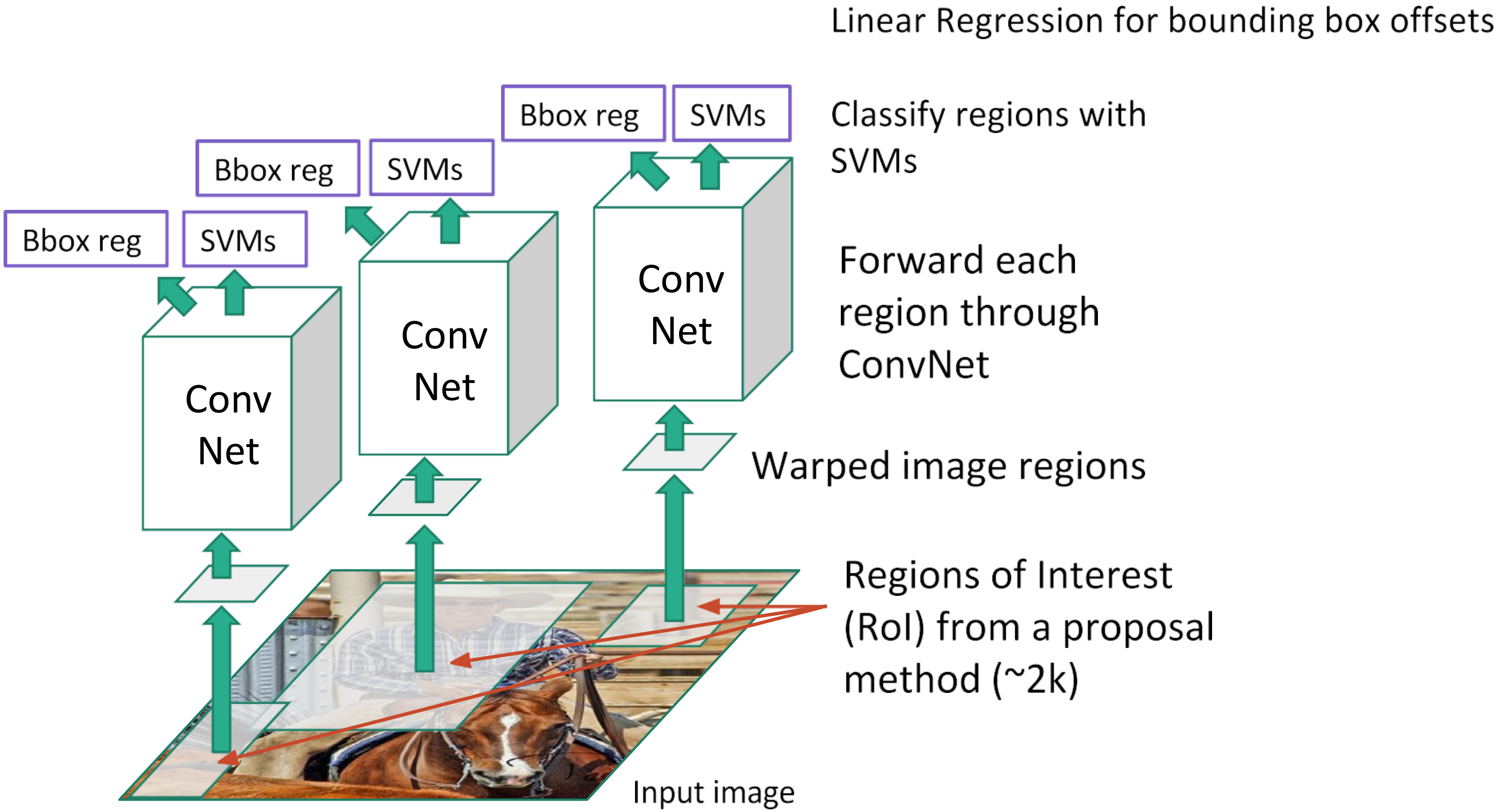
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN



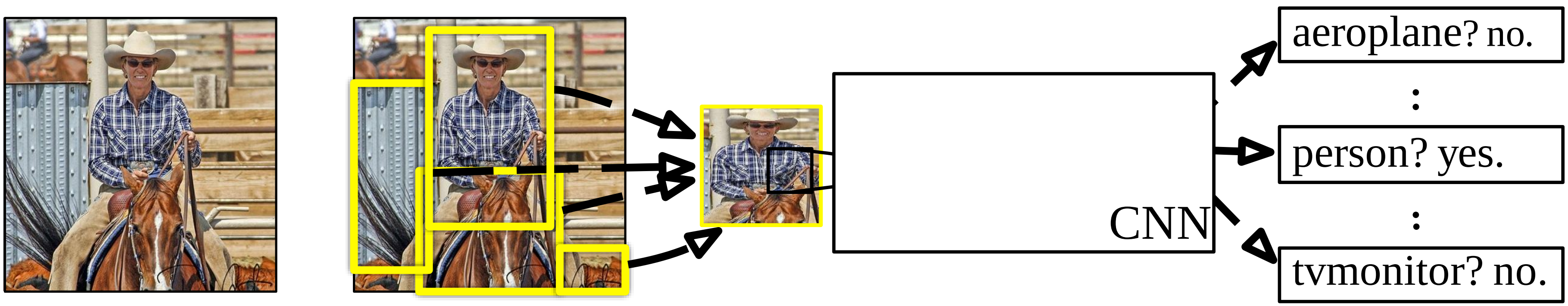
Girshick et al., “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”, CVPR 2014

R-CNN



Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN: Regions with CNN features



Input image

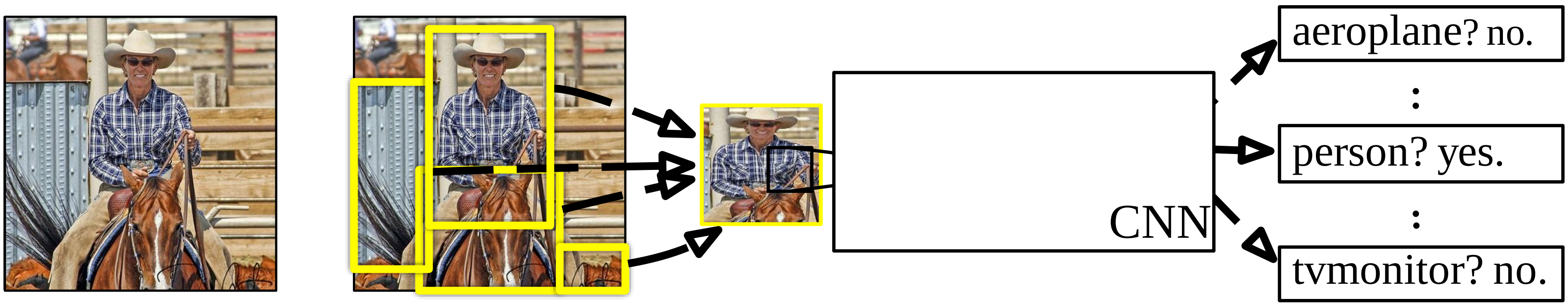
Extract region proposals (~2k / image)

Compute CNN features

Classify regions (linear SVM)

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN at test time: Step 1



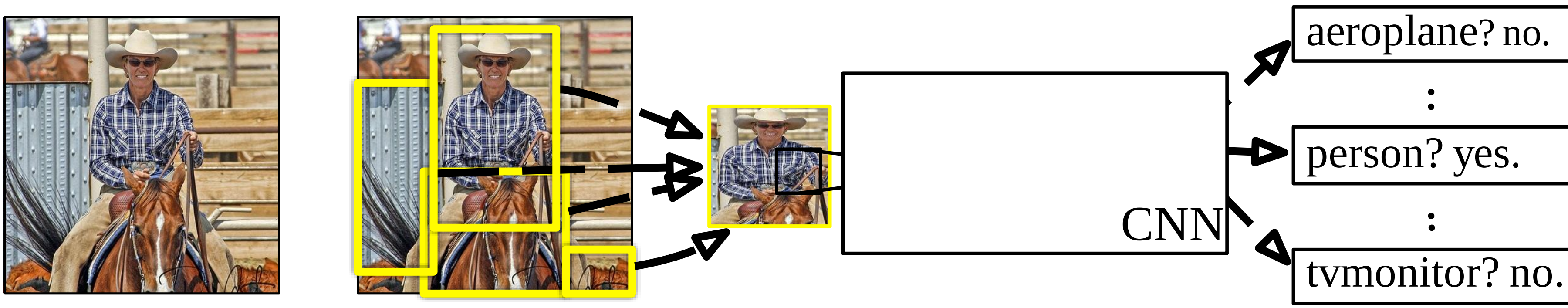
Input image → Extract region proposals (~2k / image)

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]

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

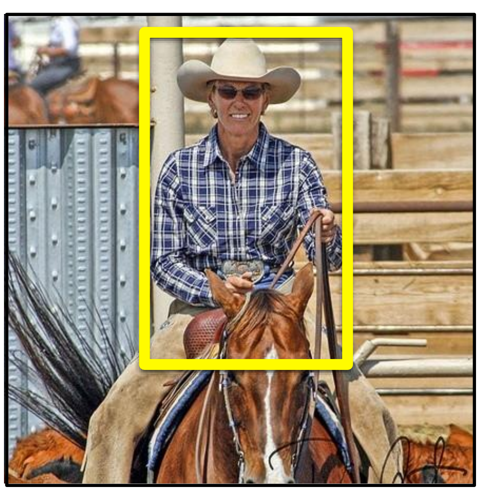
R-CNN at test time: Step 2



Input image

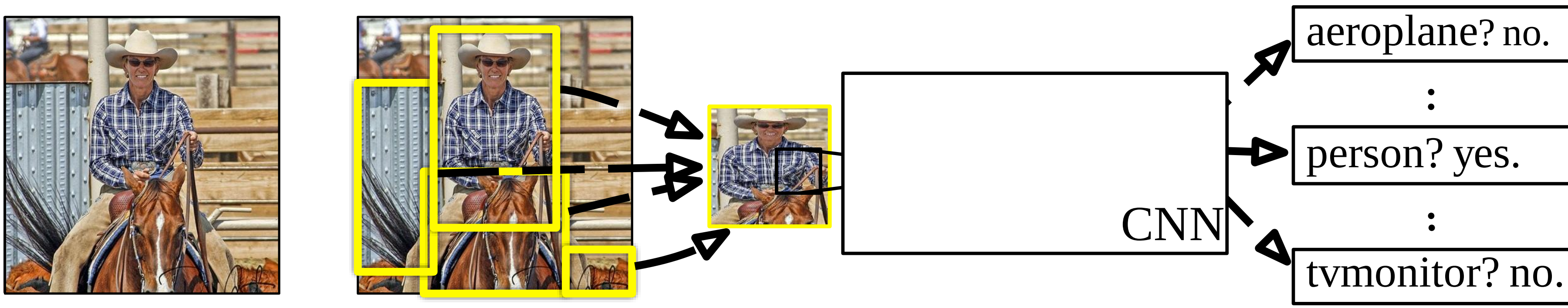
Extract region proposals (~2k / image)

Compute CNN features



Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

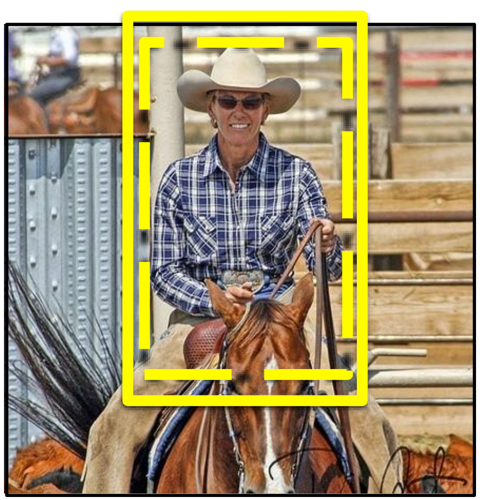
R-CNN at test time: Step 2



Input image

Extract region proposals (~2k / image)

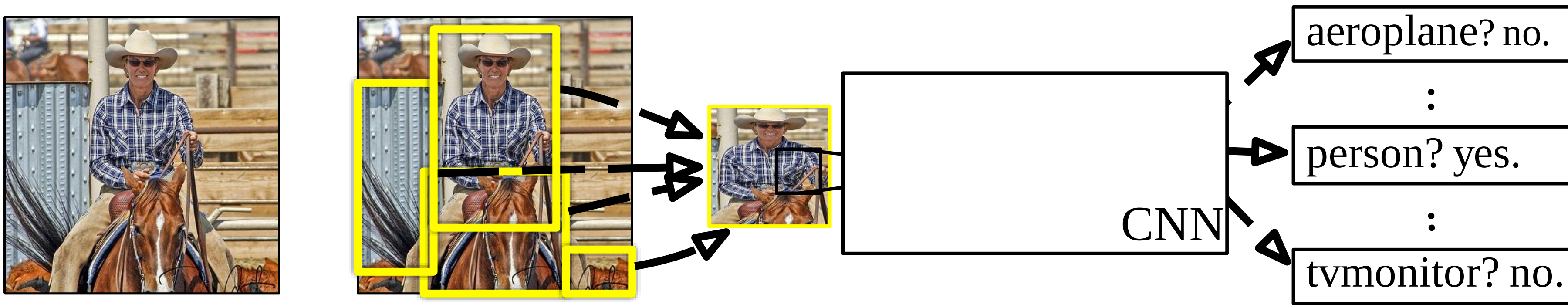
Compute CNN features



Dilate proposal

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

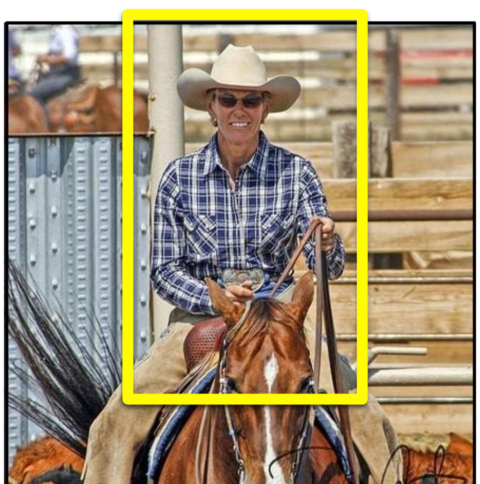
R-CNN at test time: Step 2



Input image

Extract region proposals (~2k / image)

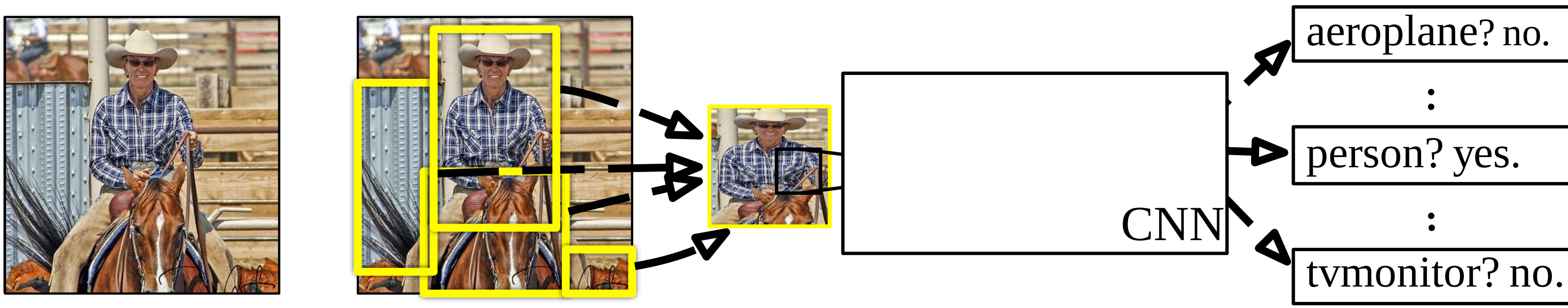
Compute CNN features



a. Crop

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

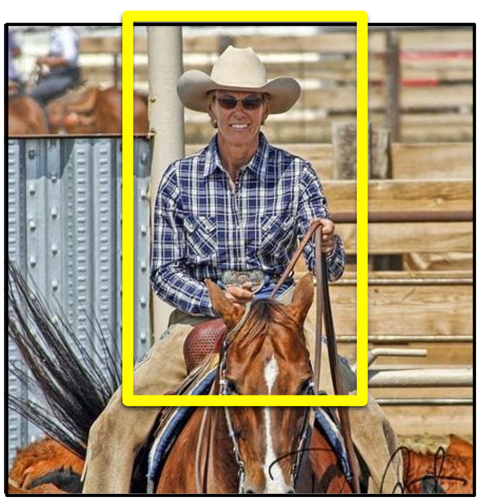
R-CNN at test time: Step 2



Input image

Extract region proposals (~2k / image)

Compute CNN features



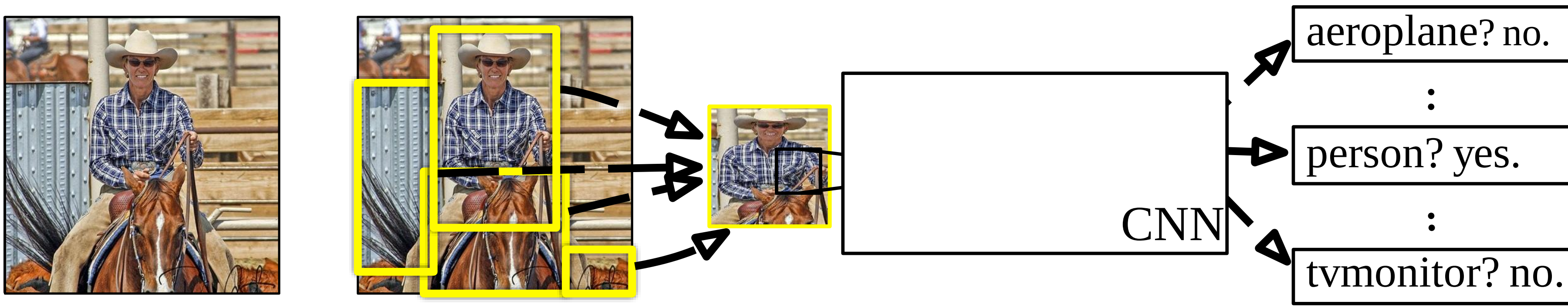
a. Crop

b. Scale (anisotropic)

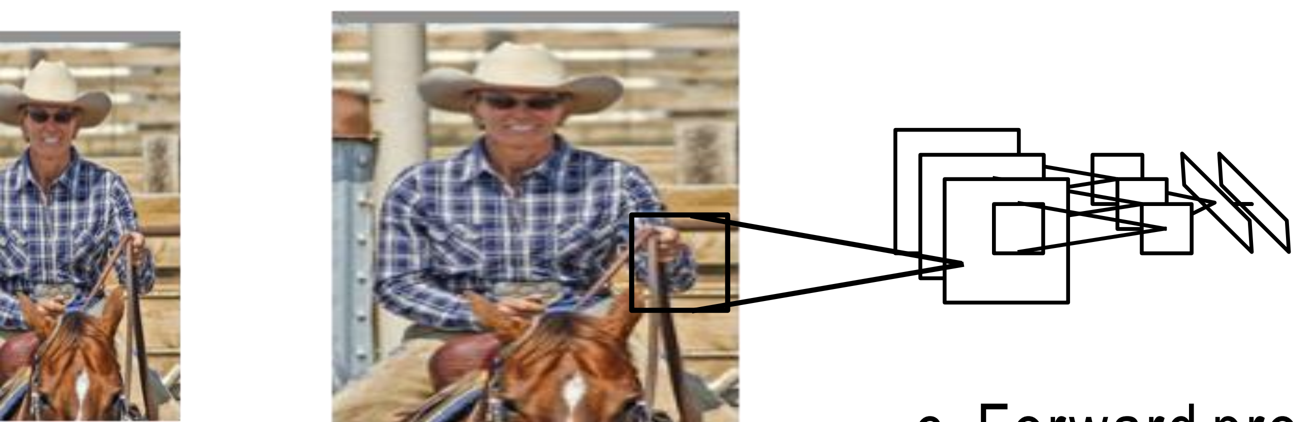
227 x 227

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN at test time: Step 2



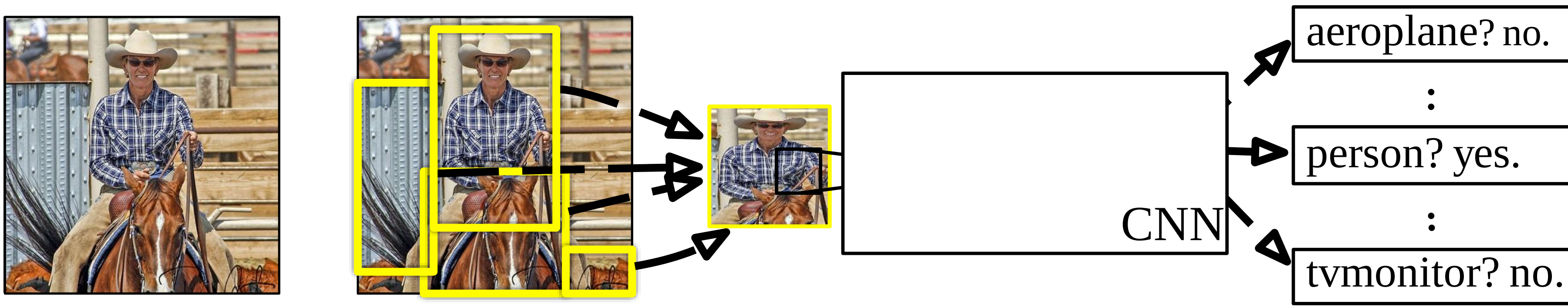
Input image | Extract region proposals (~2k / image) → Compute CNN features



. Crop | b. Scale (anisotropic) | c. Forward propagate
Output: “fc_7” features

Girshick et al., “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”, CVPR 2014

R-CNN at test time: Step 3

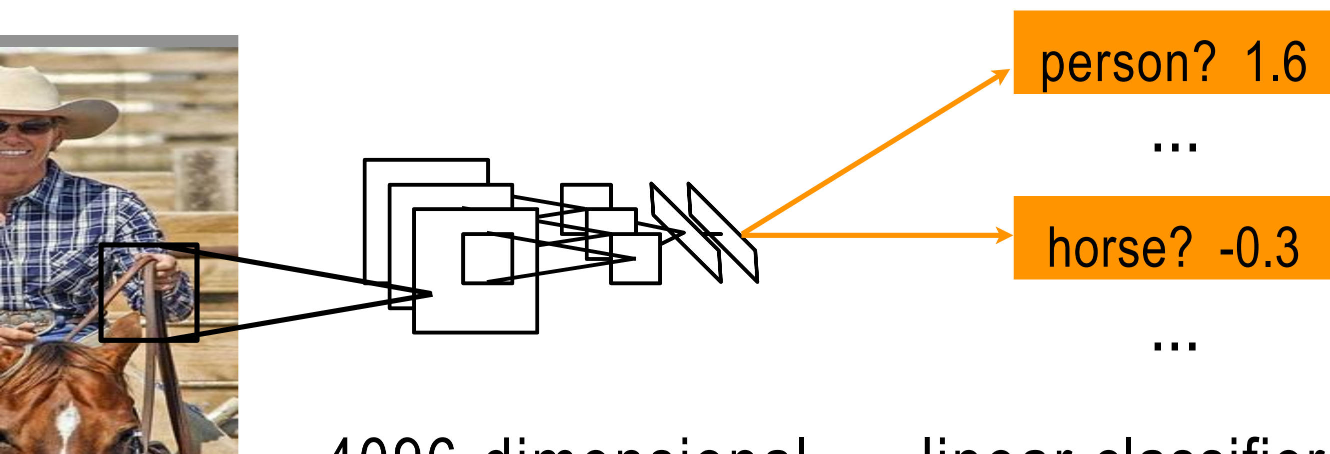


Input image

Extract region proposals (~2k / image)

Compute CNN features

Classify regions



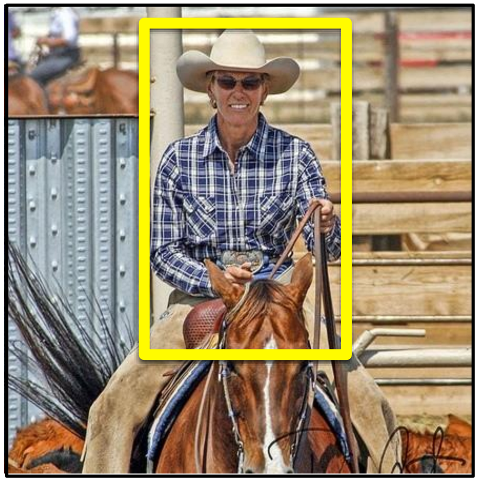
proposal

4096-dimensional fc_7 feature vector

linear classifiers (SVM or softmax)

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

Step 4: Object proposal refinement



Linear regression

on CNN features

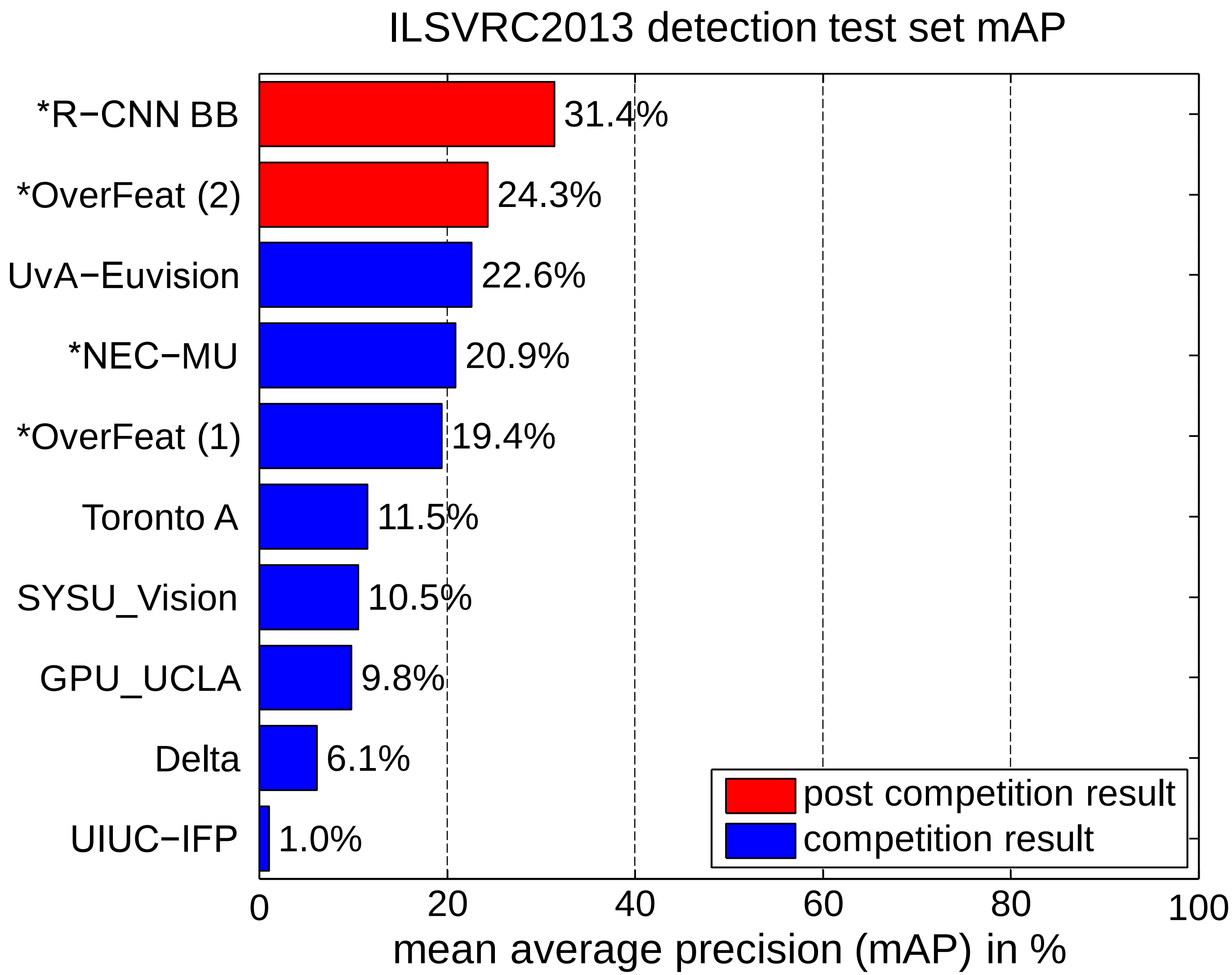
Original proposal

Predicted object bounding box

Bounding-box regression

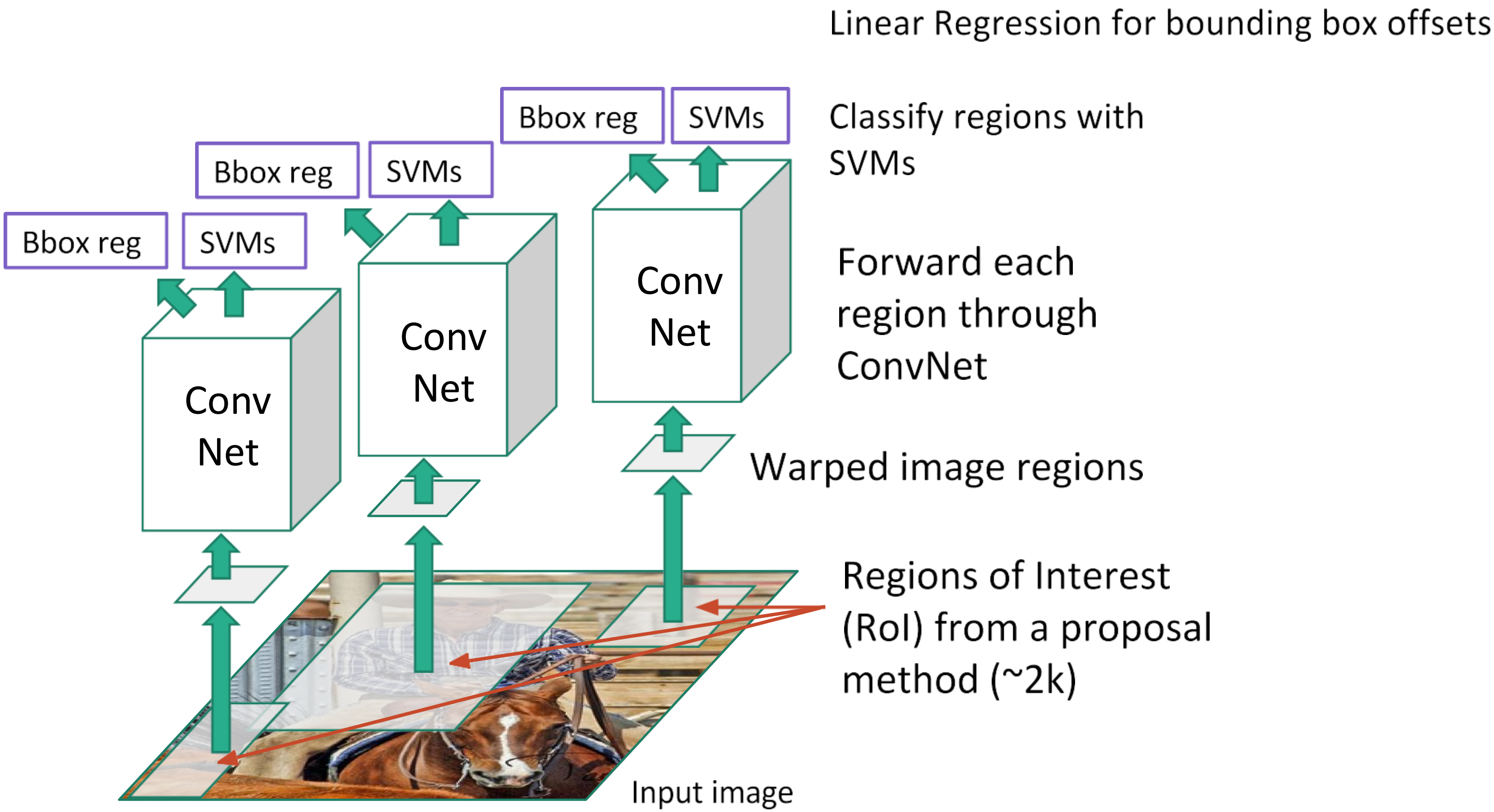
Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN on ImageNet detection



Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

R-CNN

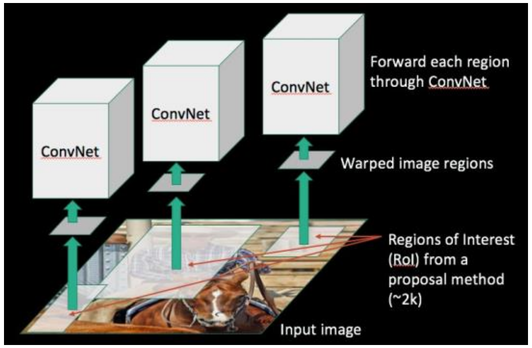


Post hoc component

Girshick et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation", CVPR 2014

What's wrong with slow R-CNN?

- 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]



~2000 ConvNet forward passes per image

Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN

- Fast test time
- One network, trained in one stage
- Higher mean average precision

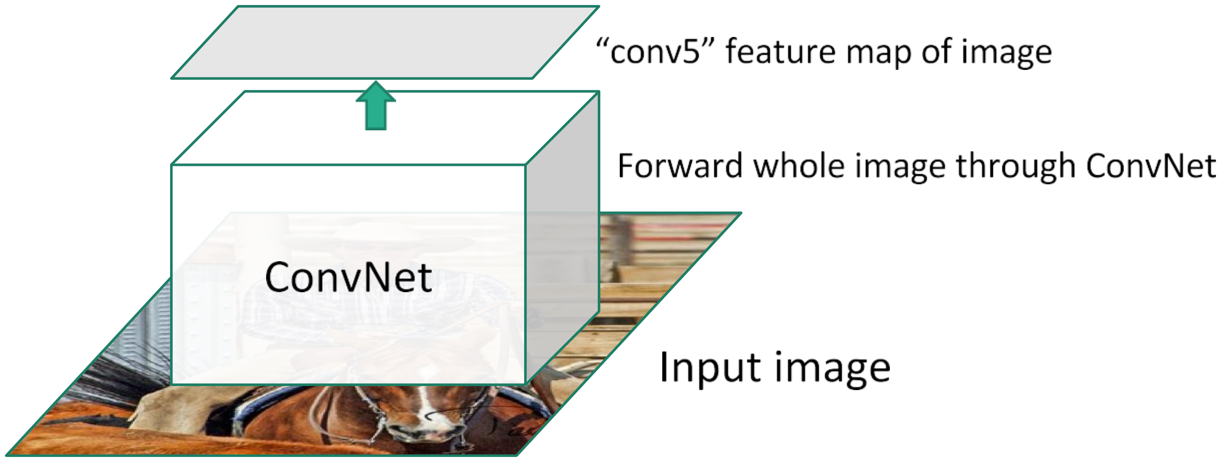
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN

Input image

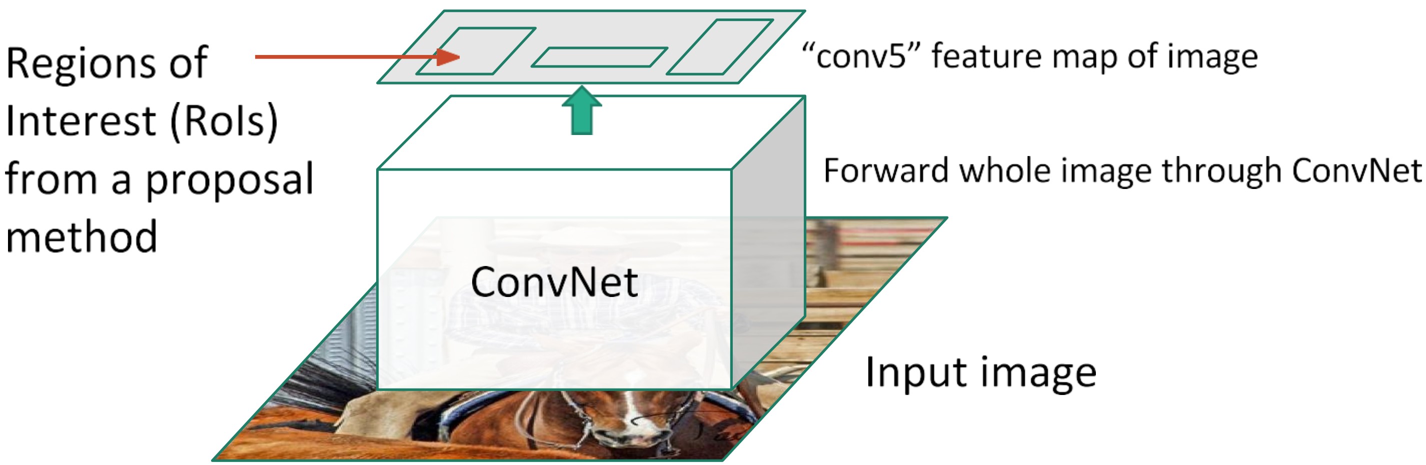
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



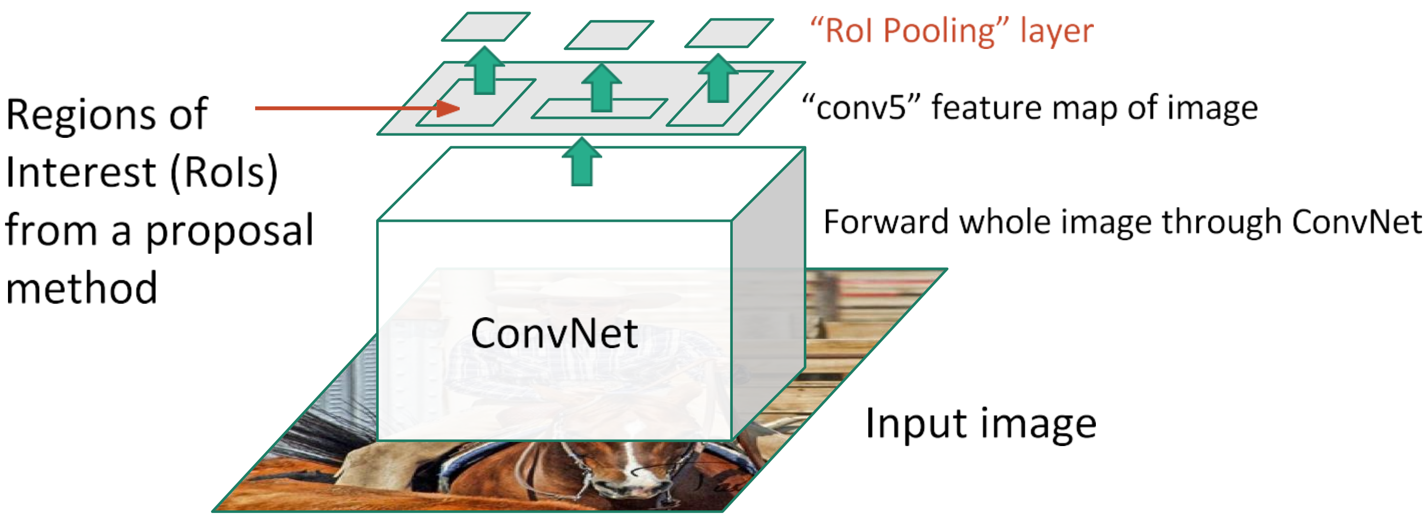
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



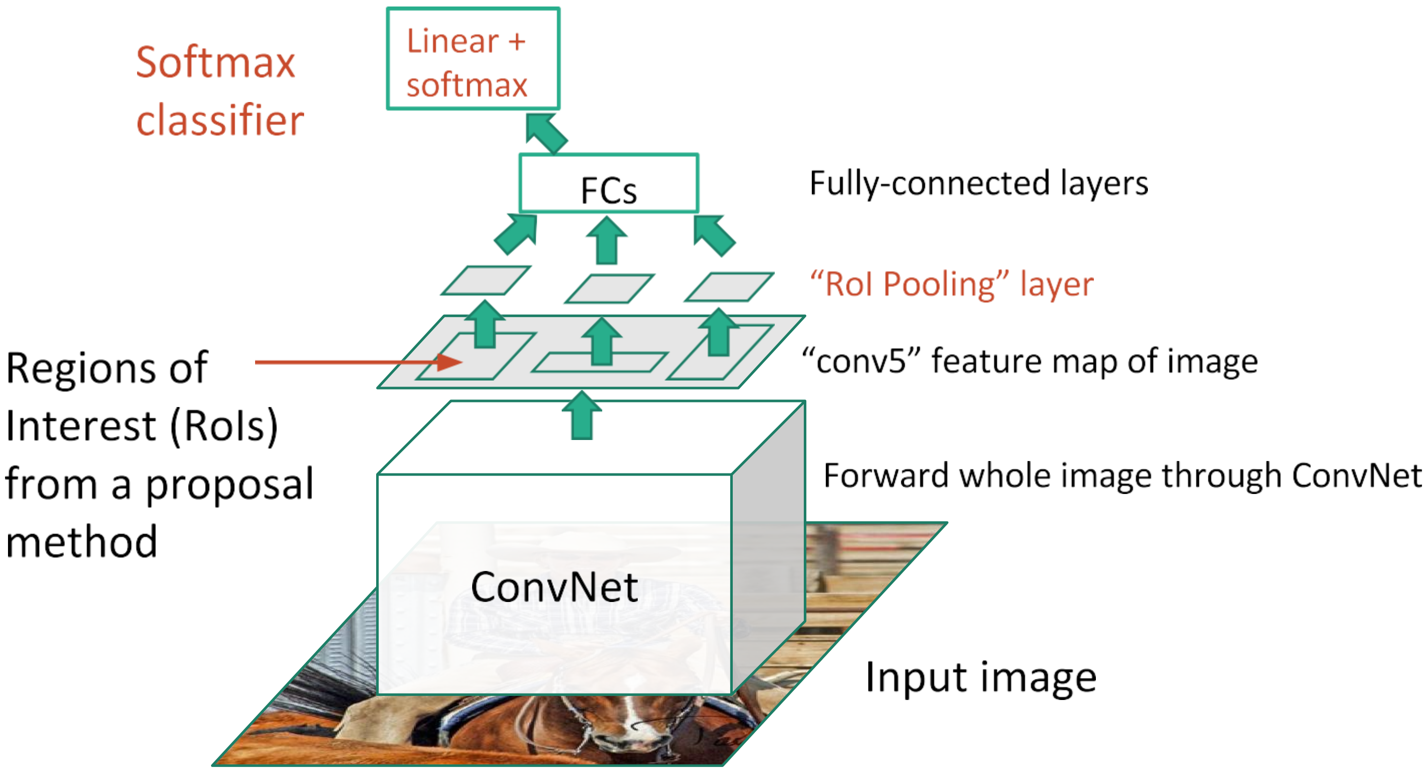
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



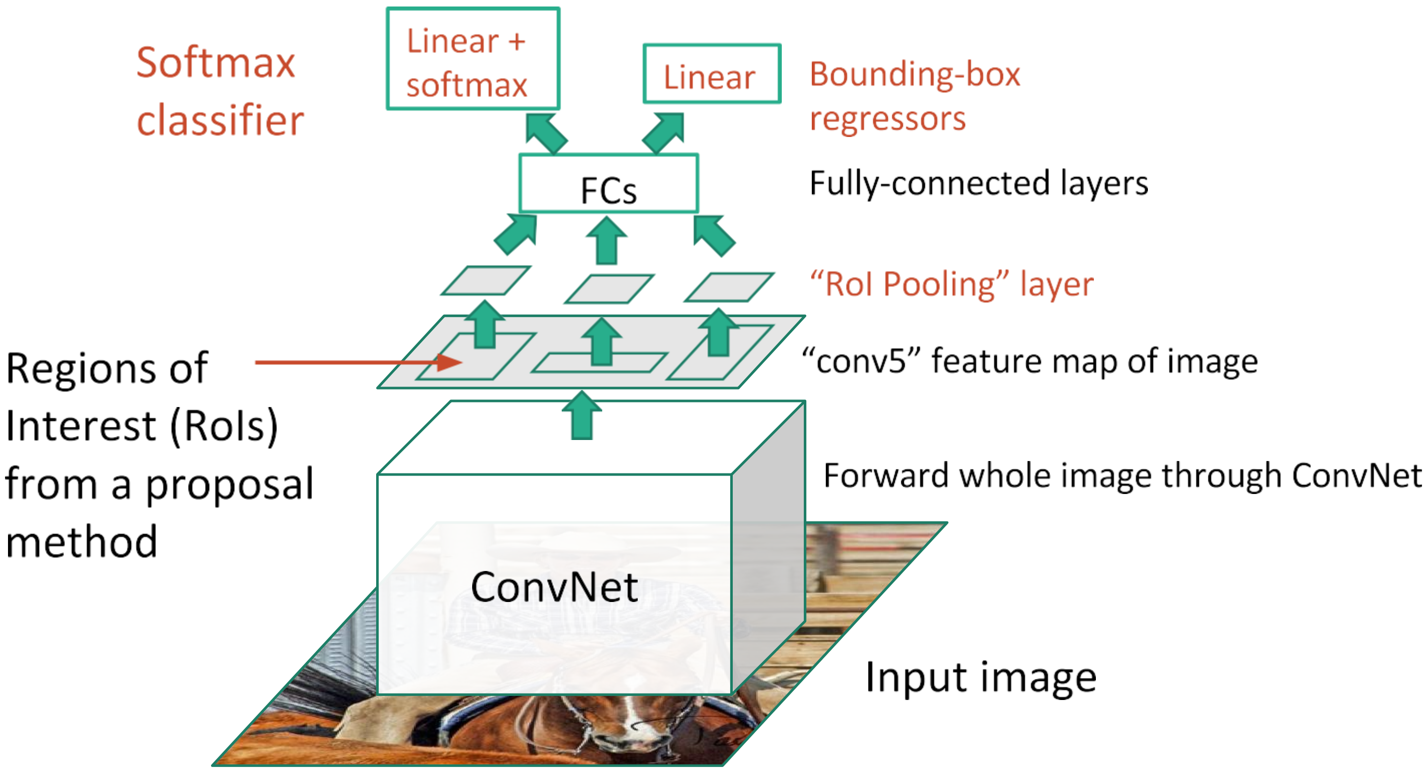
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



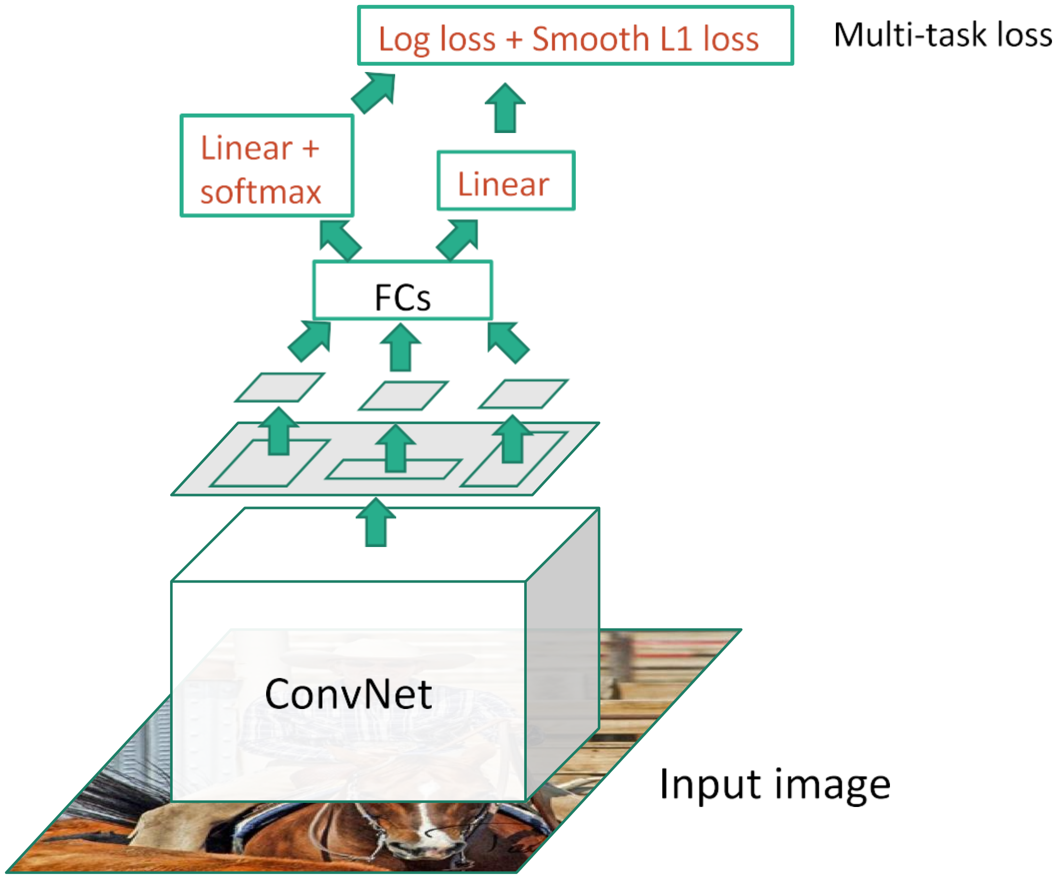
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN



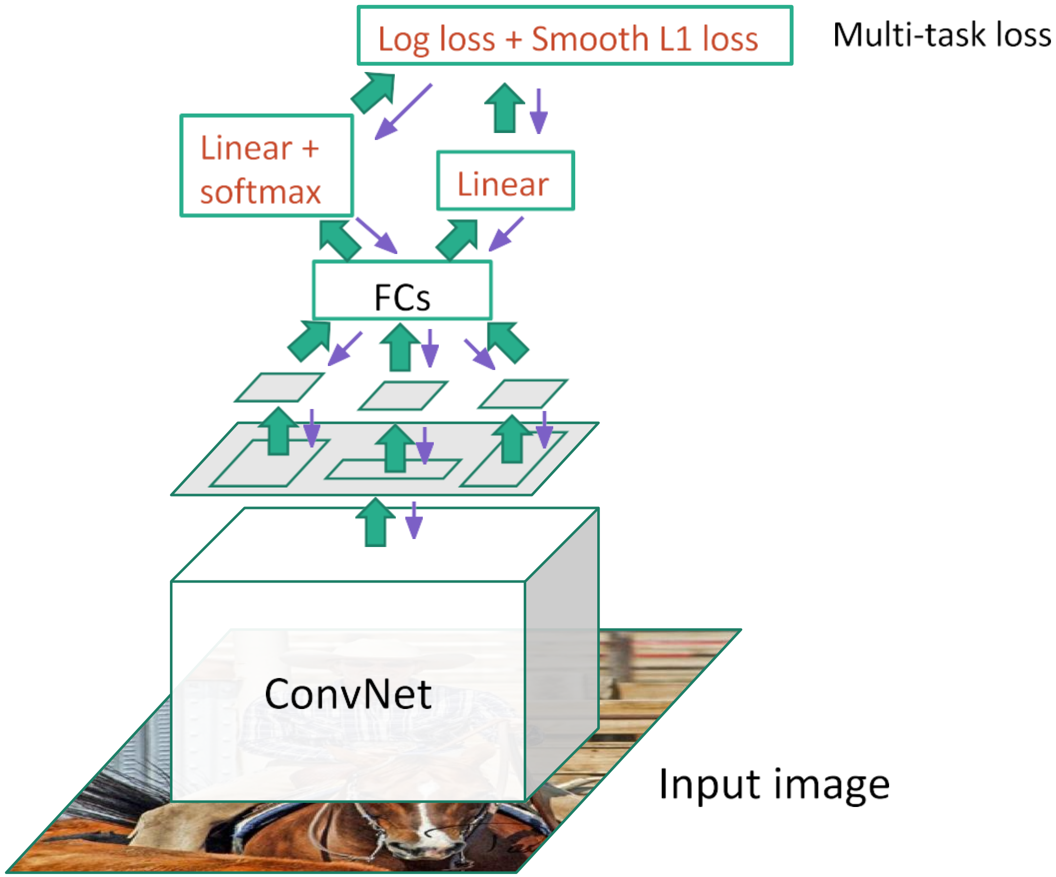
Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN (Training)



Girshick, "Fast R-CNN", ICCV 2015

Fast R-CNN (Training)



Girshick, "Fast R-CNN", ICCV 2015

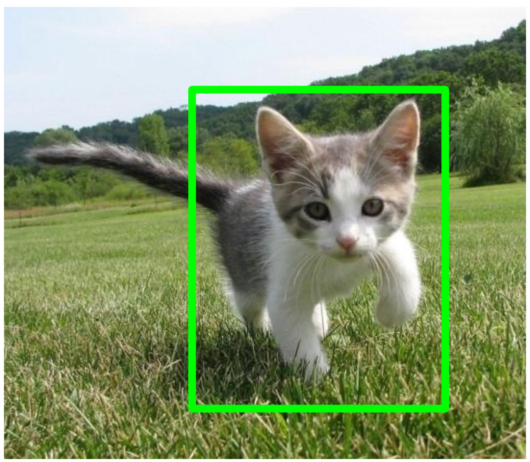
Fast R-CNN vs R-CNN

	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.

Girshick, "Fast R-CNN", ICCV 2015

Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

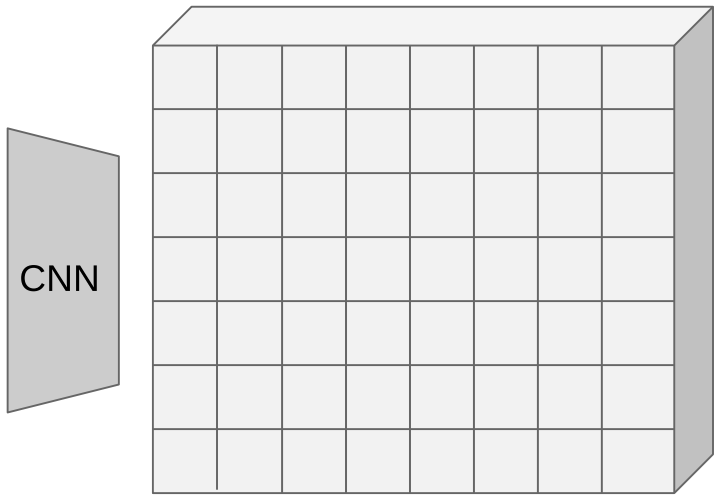
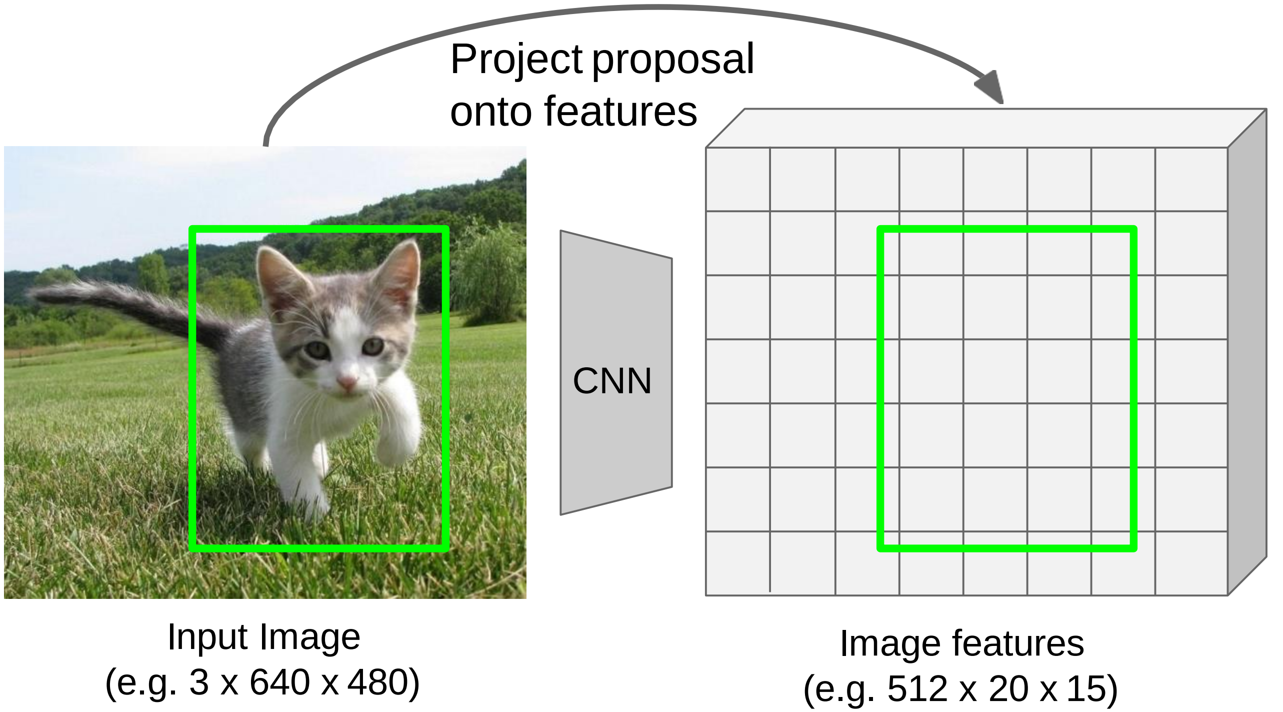


Image features
(e.g. 512 x 20 x 15)

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

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

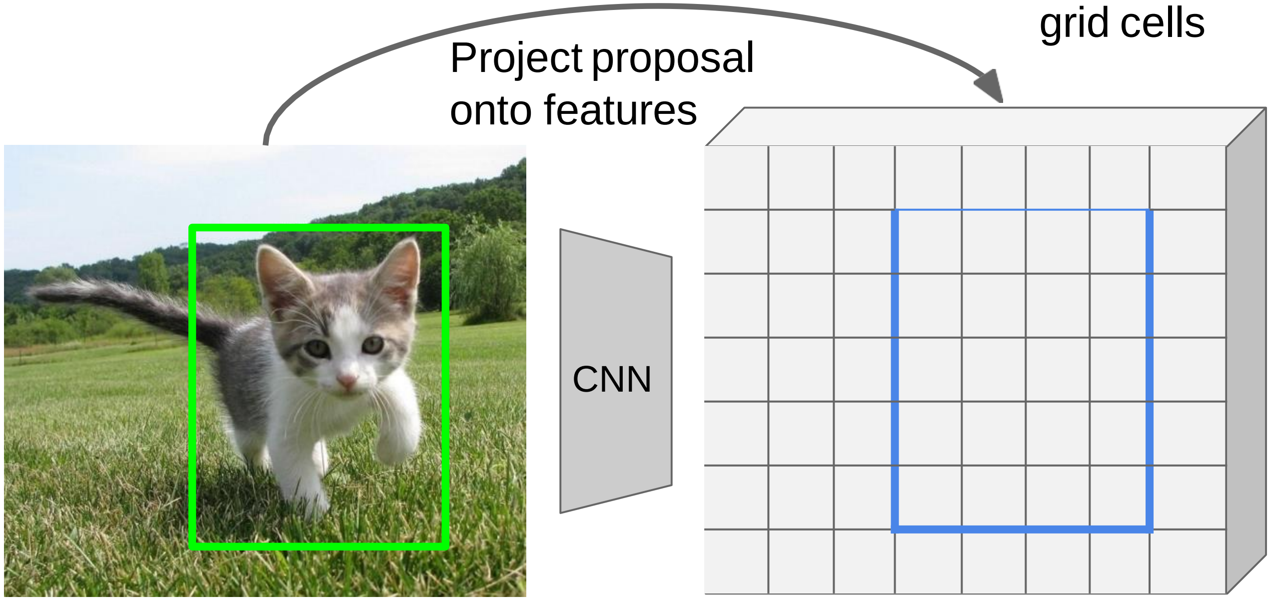
Cropping Features: RoI Pool



Girshick, “Fast R-CNN”, ICCV 2015.

Girshick, “Fast R-CNN”, ICCV 2015.

Cropping Features: RoI Pool



Input Image
(e.g. 3 x 640 x 480)

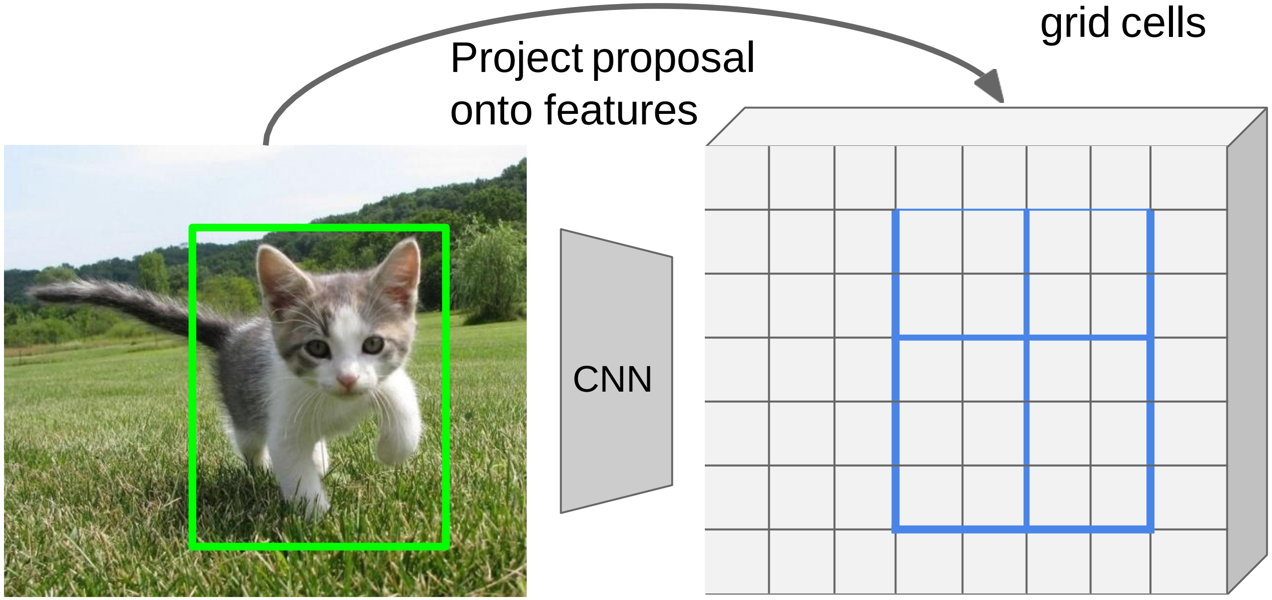
Image features
(e.g. 512 x 20 x 15)

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

Cropping Features: RoI Pool

"Snap" to
grid cells

Divide into 2x2
grid of (roughly)
equal subregions

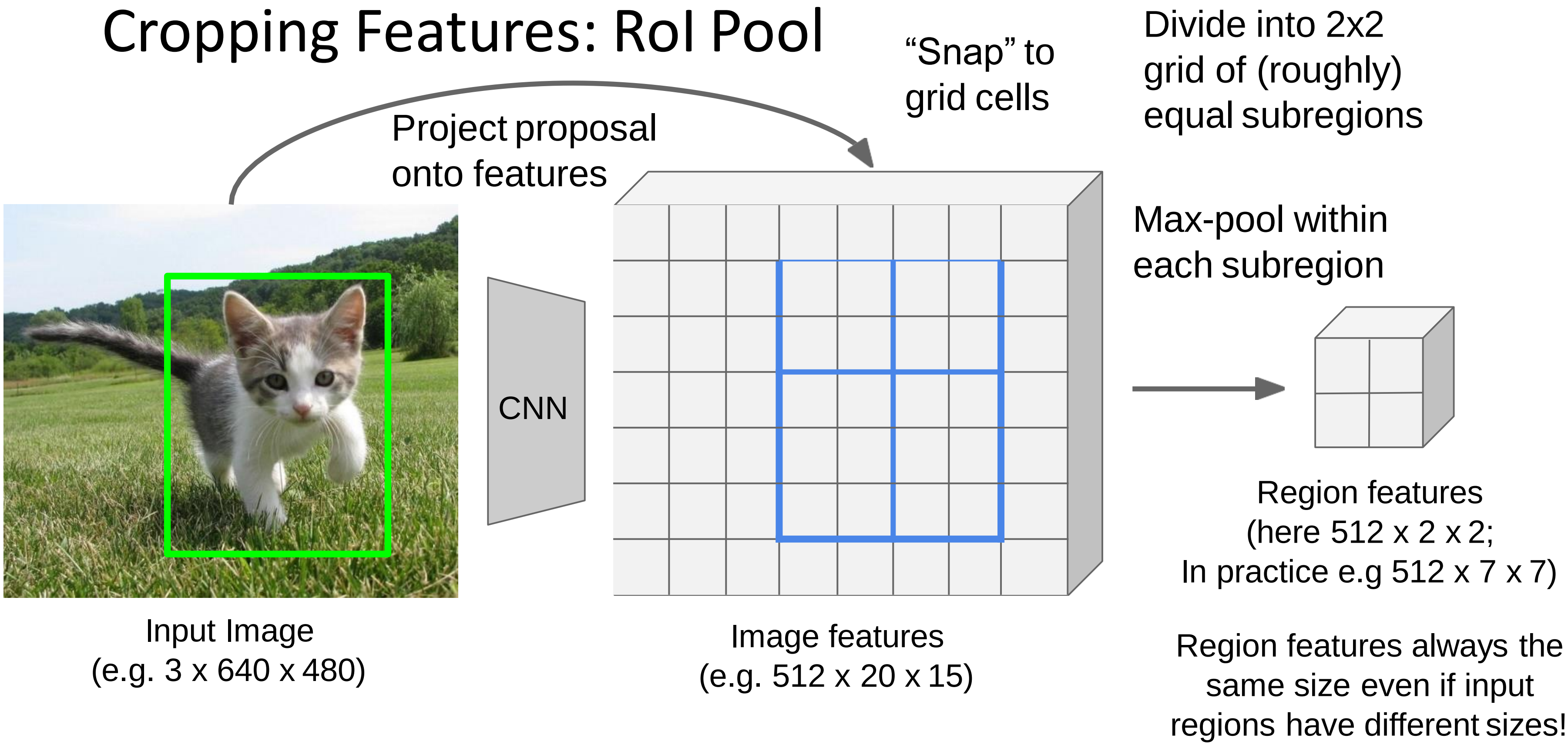


Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

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

Cropping Features: RoI Pool

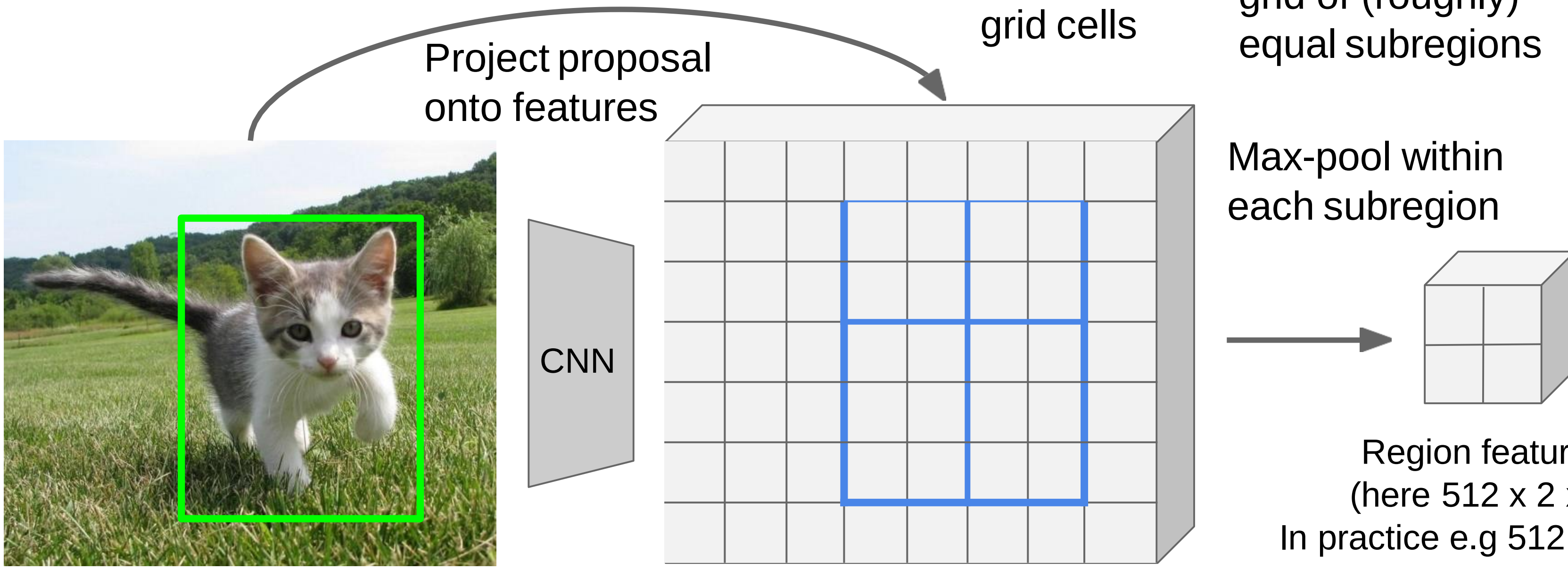


Girshick, "Fast R-CNN", ICCV 2015.

Cropping Features: RoI Pool

"Snap" to grid cells

Divide into 2x2 grid of (roughly) equal subregions



Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

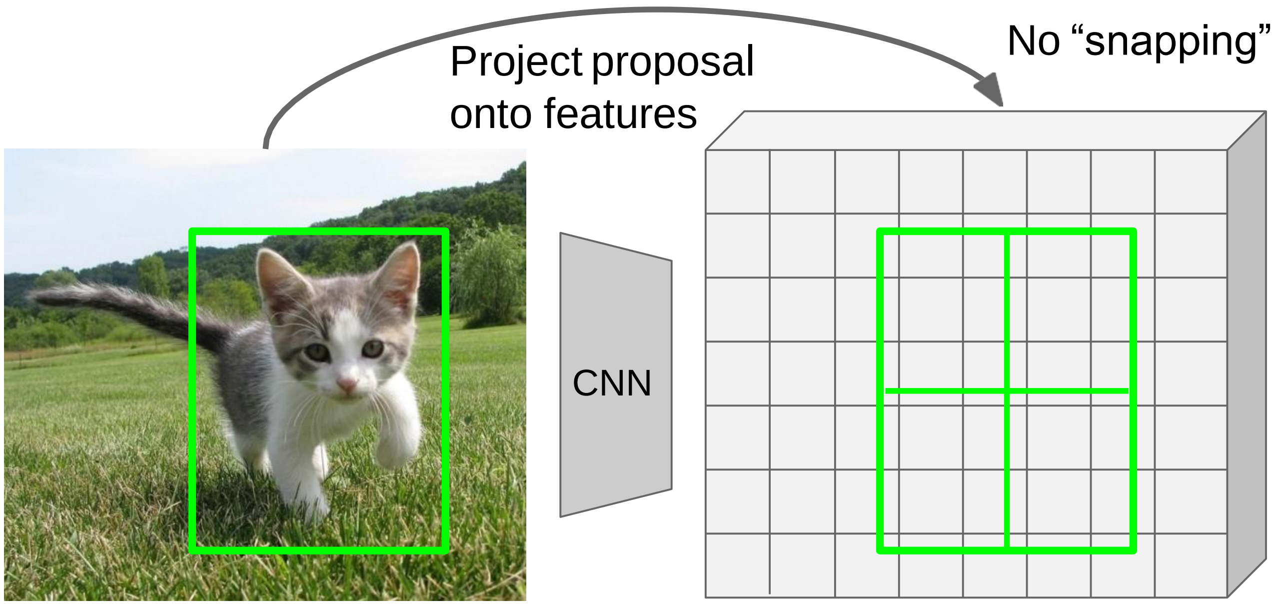
Region features
(here 512 x 2 x 2;
In practice e.g 512 x 7 x 7)

Region features always the same size even if input regions have different sizes!

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

Problem: Region features slightly misaligned

Cropping Features: RoI Align



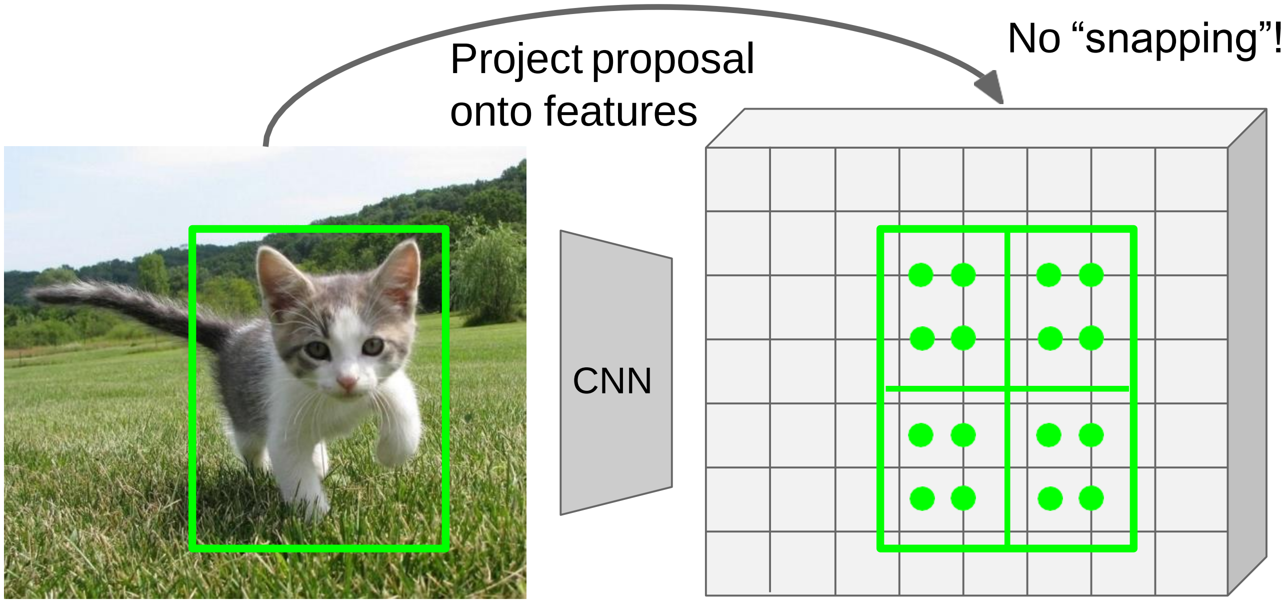
Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

He et al, “Mask R-CNN”, ICCV 2017

Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



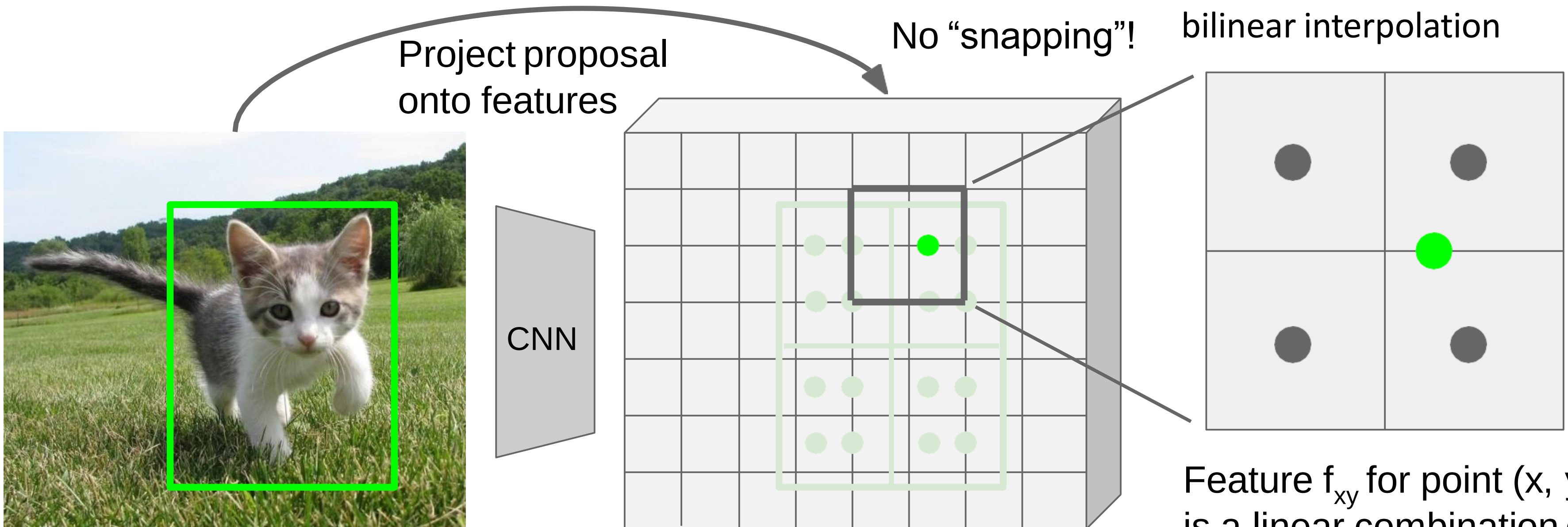
Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

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

Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



Input Image
(e.g. 3 x 640 x 480)

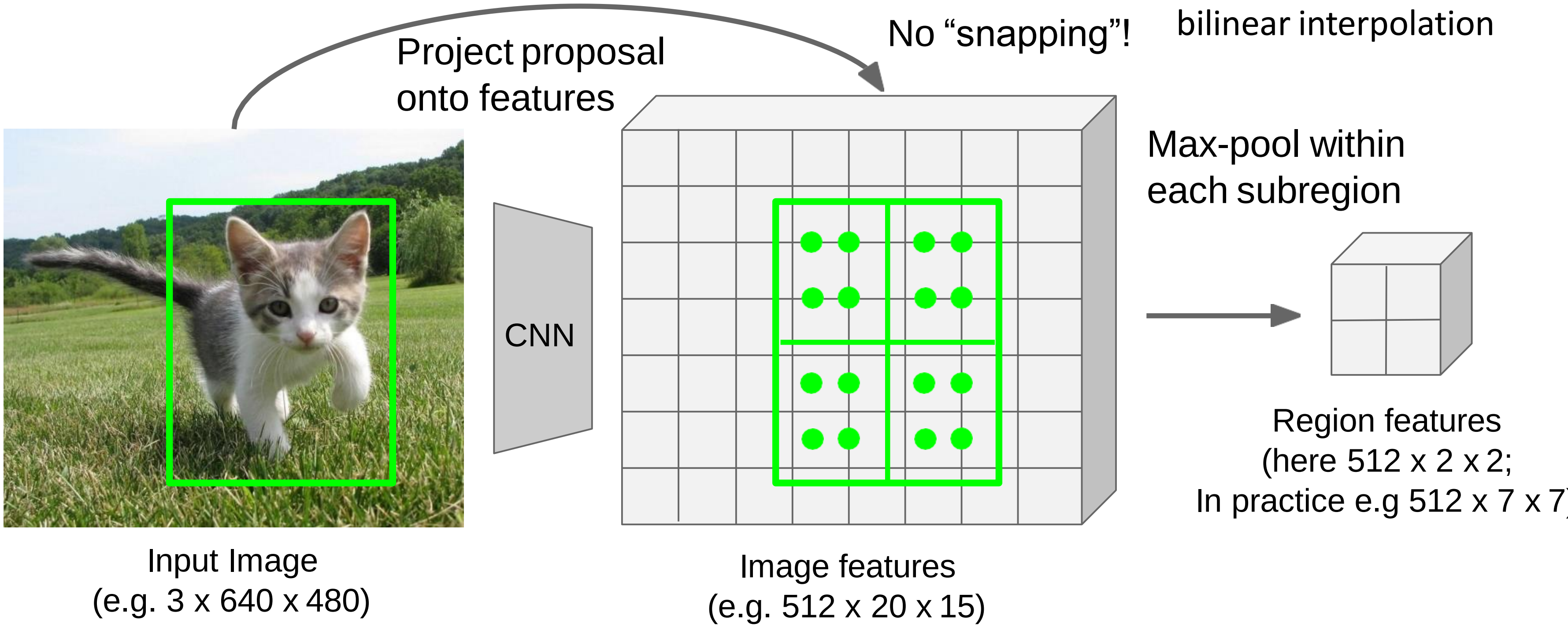
Image features
(e.g. 512 x 20 x 15)

Feature f_{xy} for point (x, y) is a linear combination of features at its four neighboring grid cells

He et al, “Mask R-CNN”, ICCV 2017

Cropping Features: RoI Align

Sample at regular points in each subregion using bilinear interpolation



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

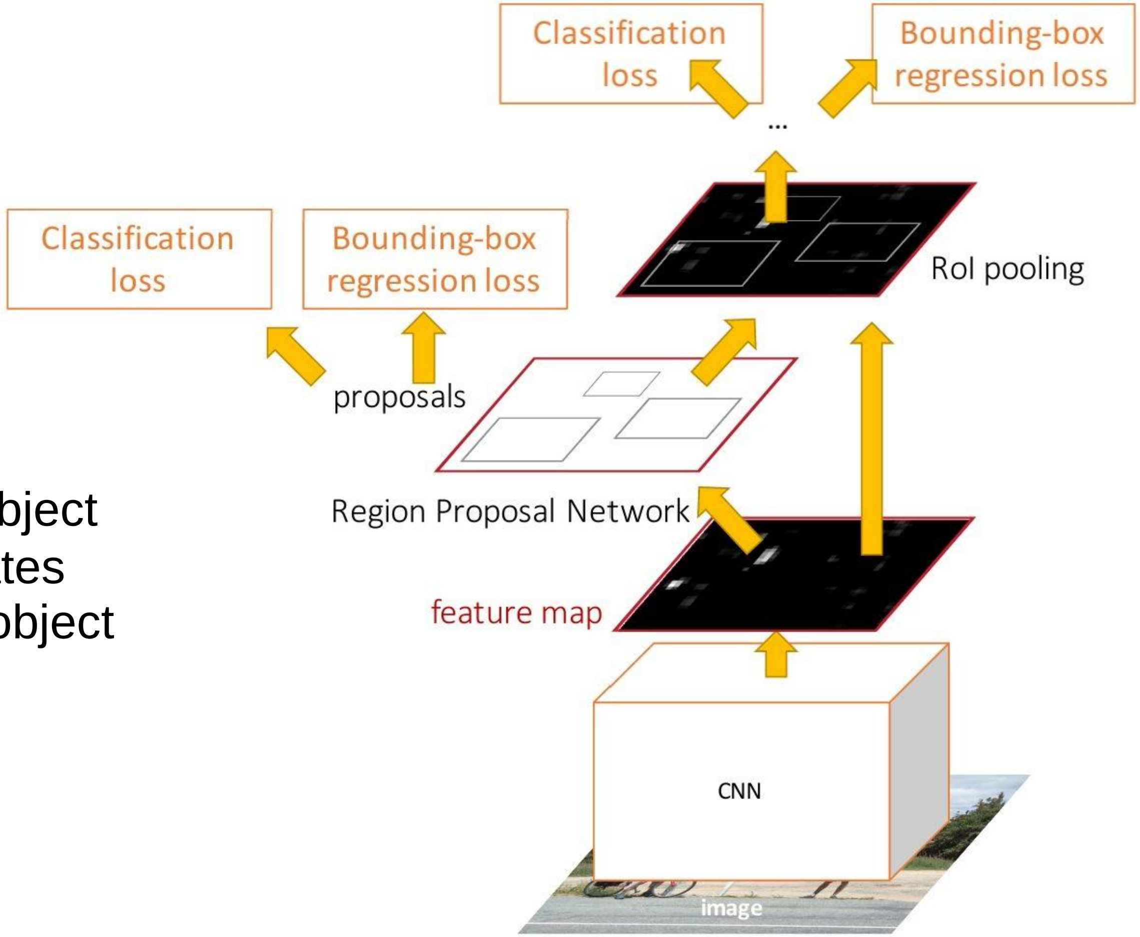
Faster R-CNN

Make CNN do proposals!

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

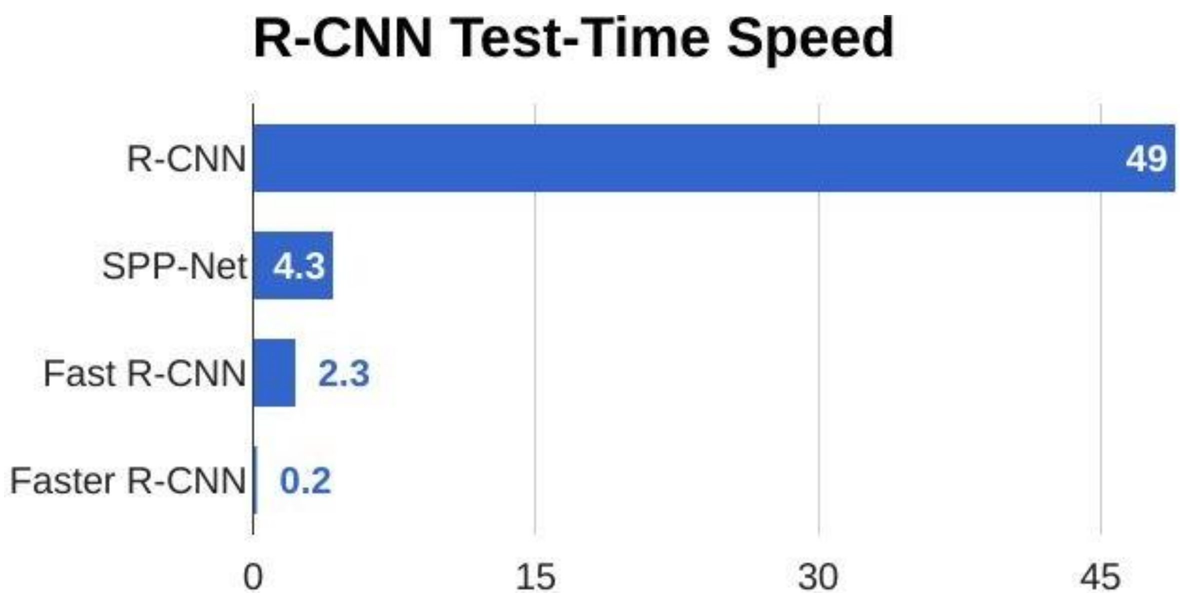
Jointly train with 4 losses:

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

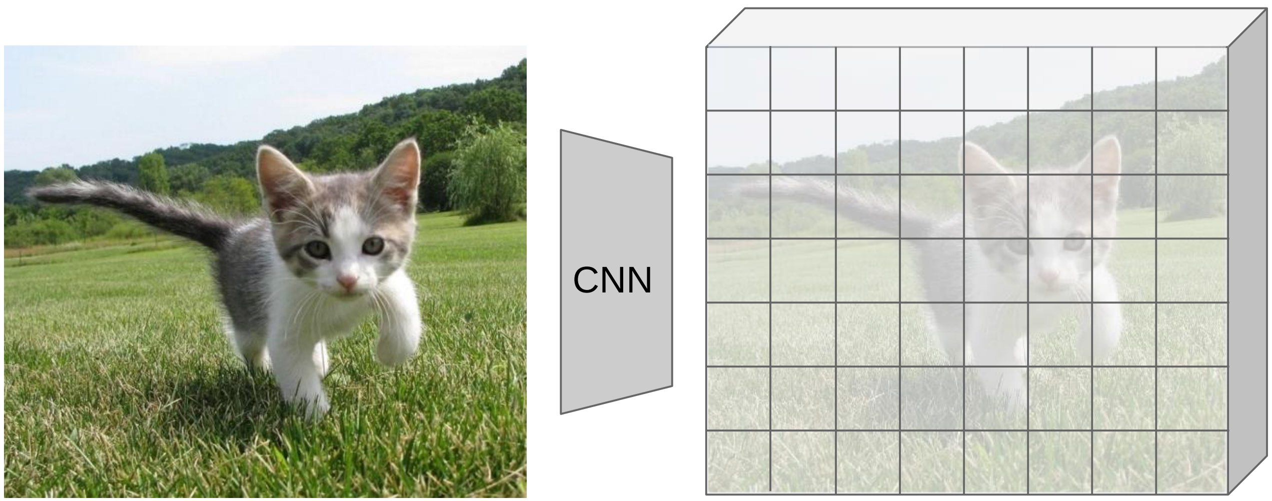


Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015

Faster R-CNN


R-CNN Test-Time Speed
R-CNN
49
SPP-Net
4.3
Fast R-CNN
2.3
Faster R-CNN
0.2
0
15
30
45

Region Proposal Network

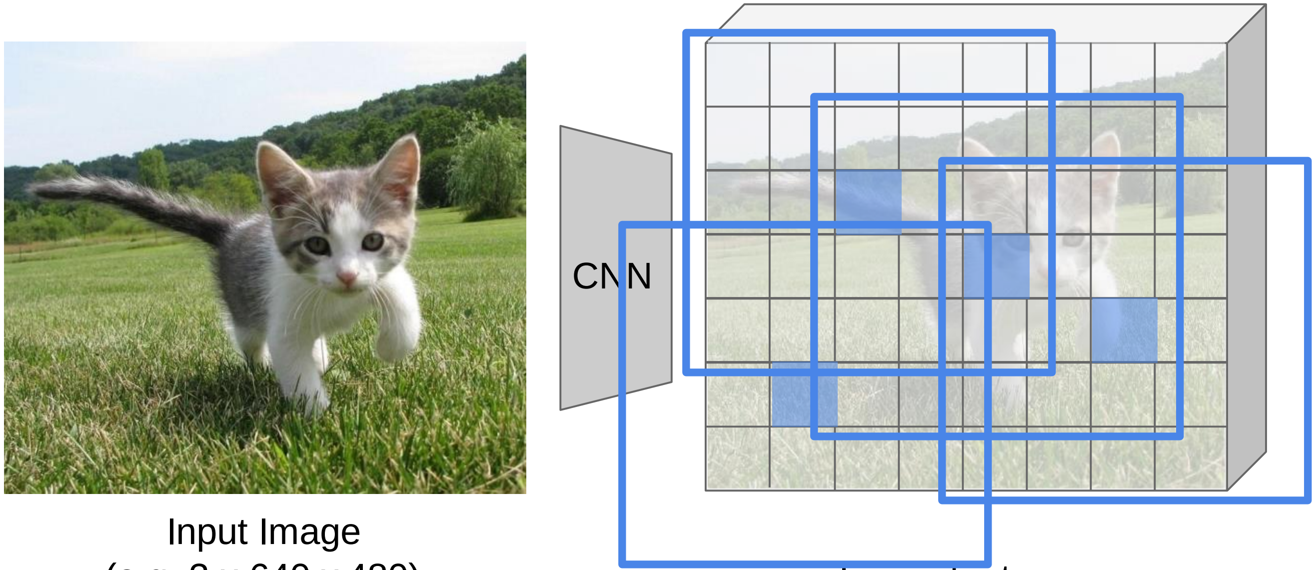


Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map

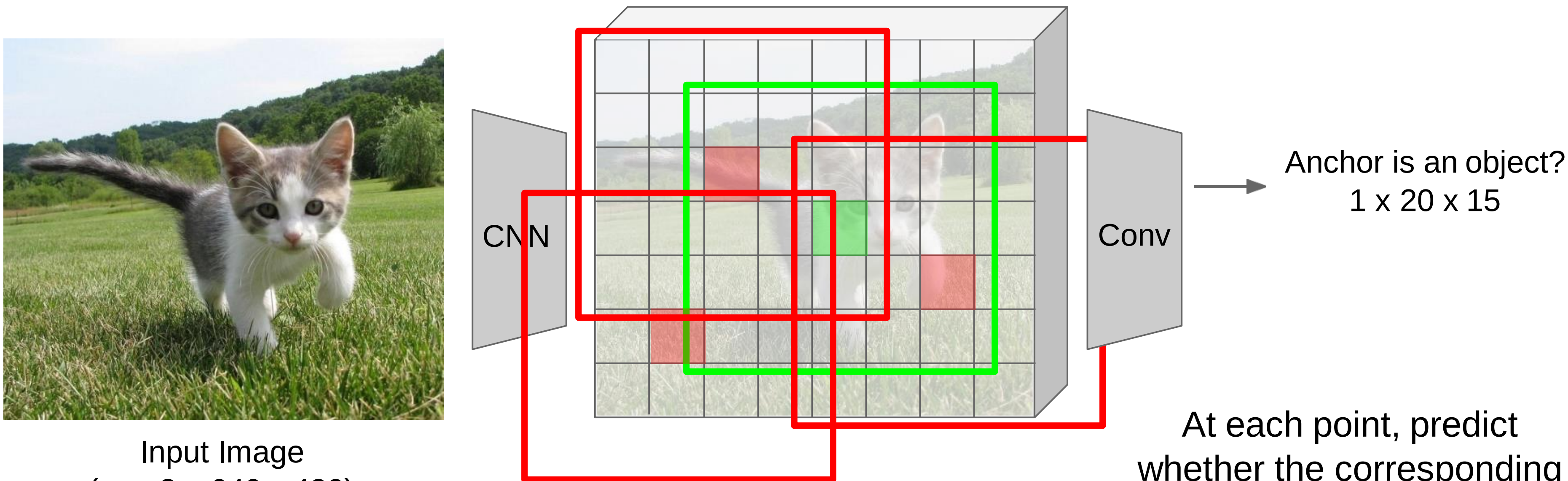


Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map



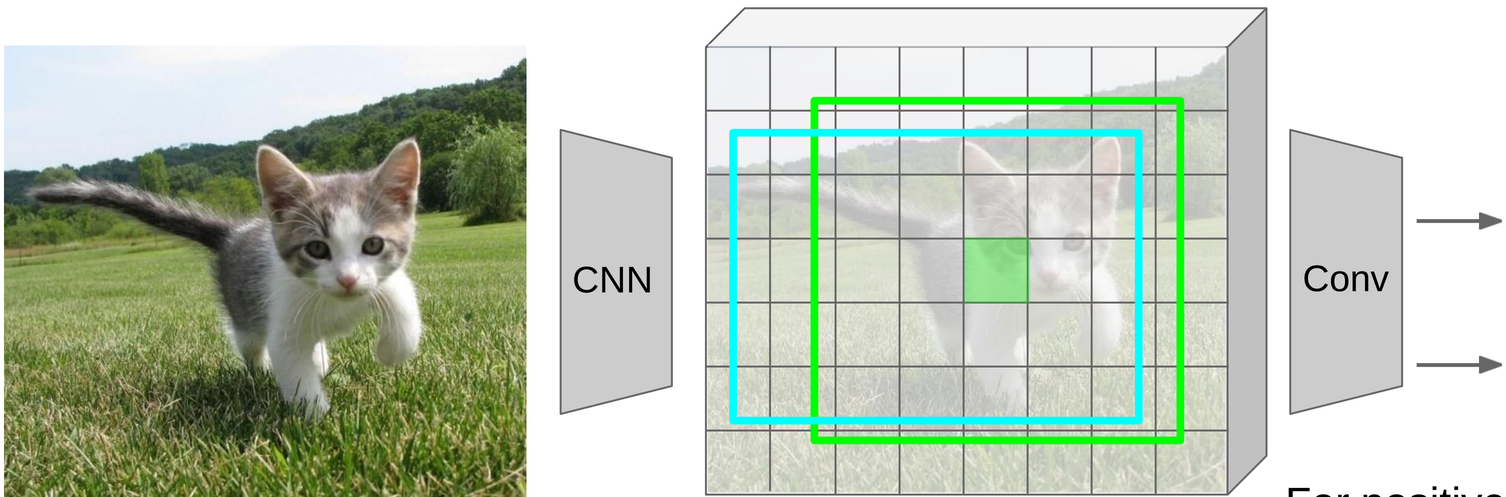
Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

At each point, predict whether the corresponding anchor contains an object (per-pixel logistic regression)

Region Proposal Network

Imagine an **anchor box** of fixed size at each point in the feature map



Anchor is an object?
1 x 20 x 15

Box transforms
4 x 20 x 15

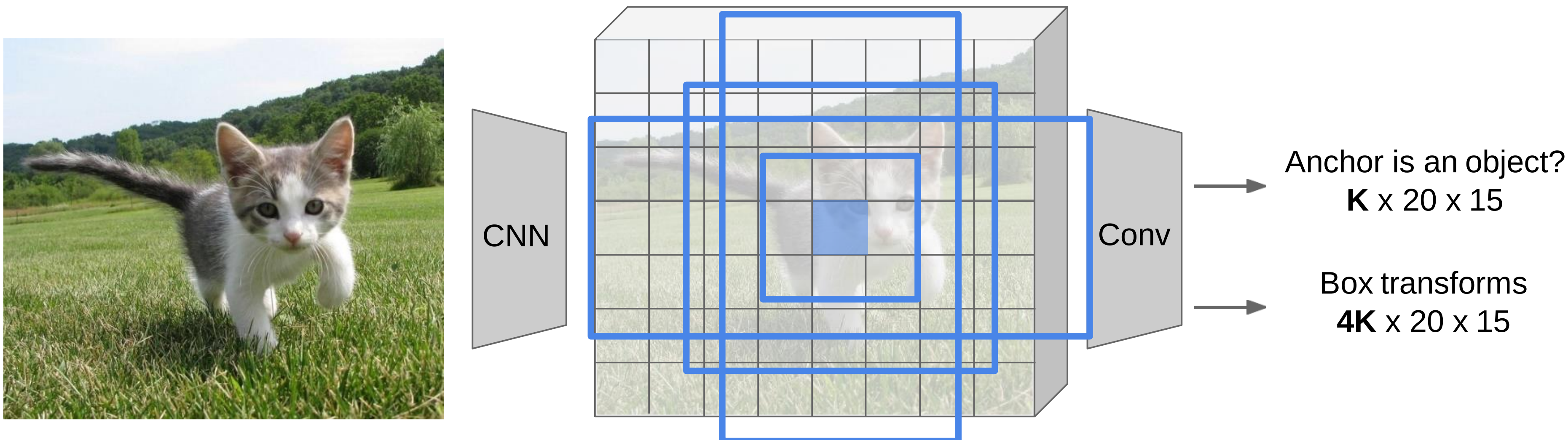
Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

For positive boxes, also predict a transformation from the anchor to the ground-truth box (regress 4 numbers per pixel)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point

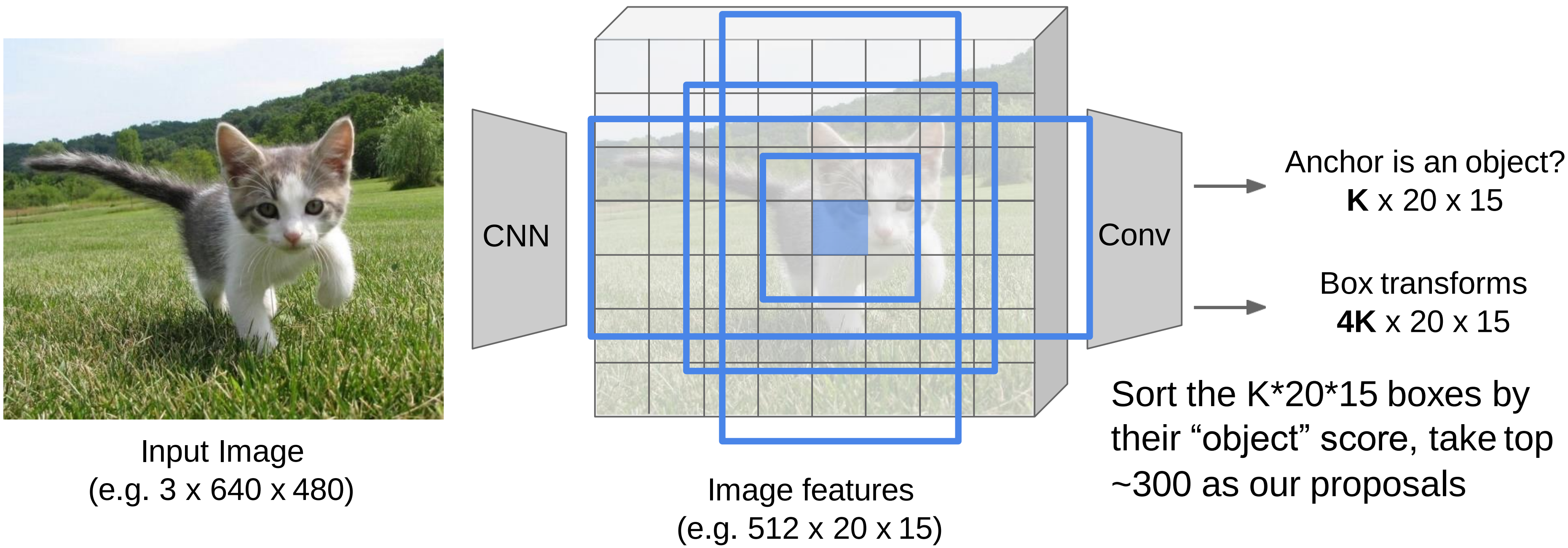


Input Image
(e.g. 3 x 640 x 480)

Image features
(e.g. 512 x 20 x 15)

Region Proposal Network

In practice use K different anchor boxes of different size / scale at each point



Sort the K*20*15 boxes by their “object” score, take top ~300 as our proposals

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

⅓ Mile, 1760 feet

Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

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

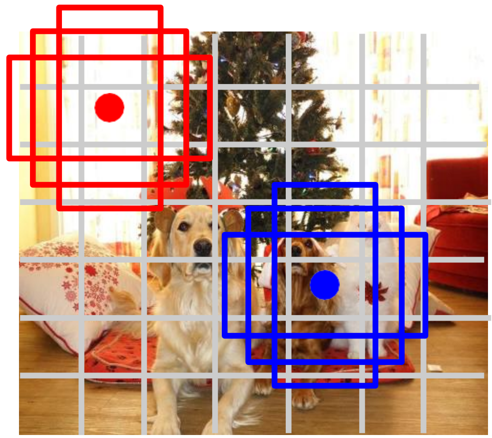
Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Plan for this lecture

- Fully supervised detection
  - Pre-CNN: Deformable part models
  - Detection with region proposals: R-CNN, Fast/er R-CNN
  - Detection without region proposals: YOLO
  - Semantic and instance segmentation: FCN, Mask R-CNN
- Weak or out-of-domain supervision
  - Weakly supervised object detection
  - Domain adaptation

Detection without Proposals: YOLO / SSD

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

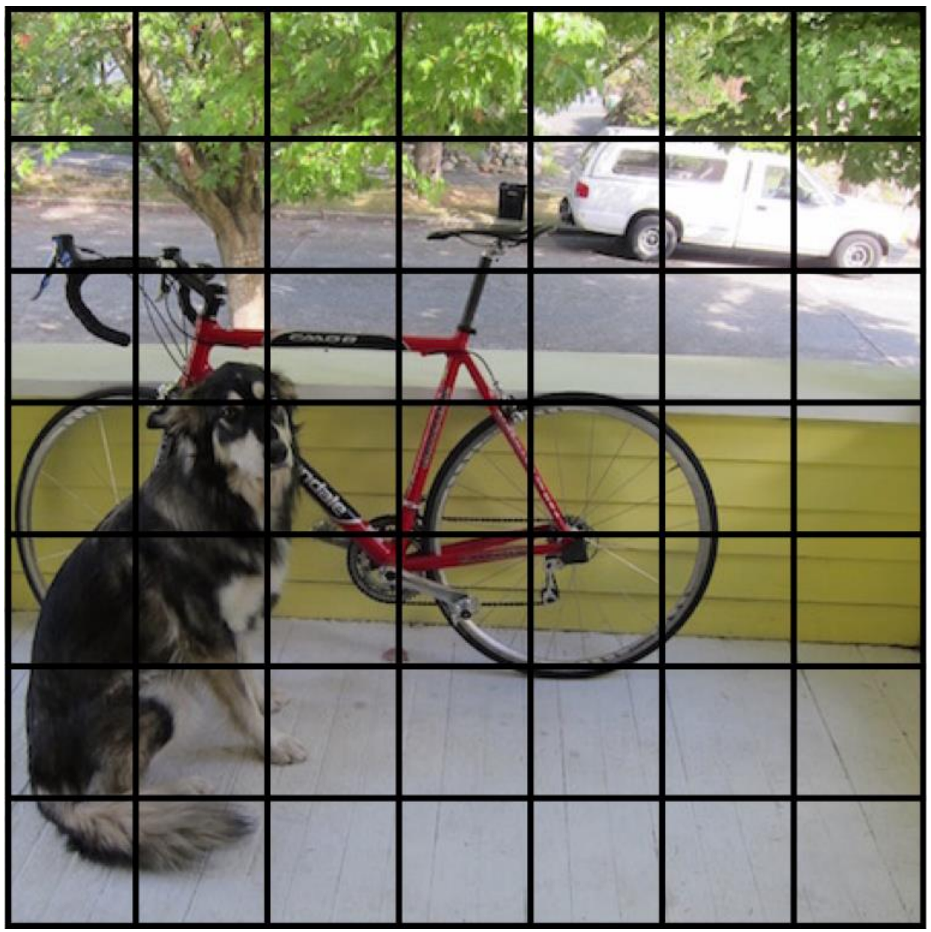
Within each grid cell:
- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

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

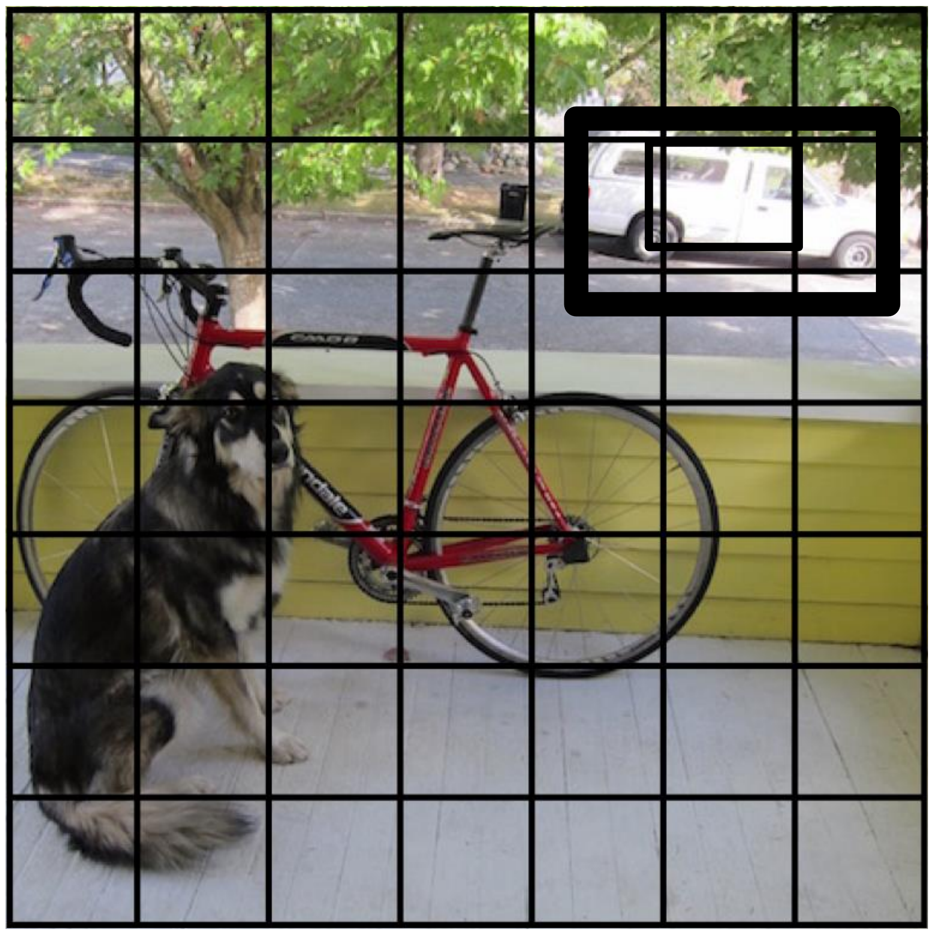
Slide by: Justin Johnson

Split the image into a grid



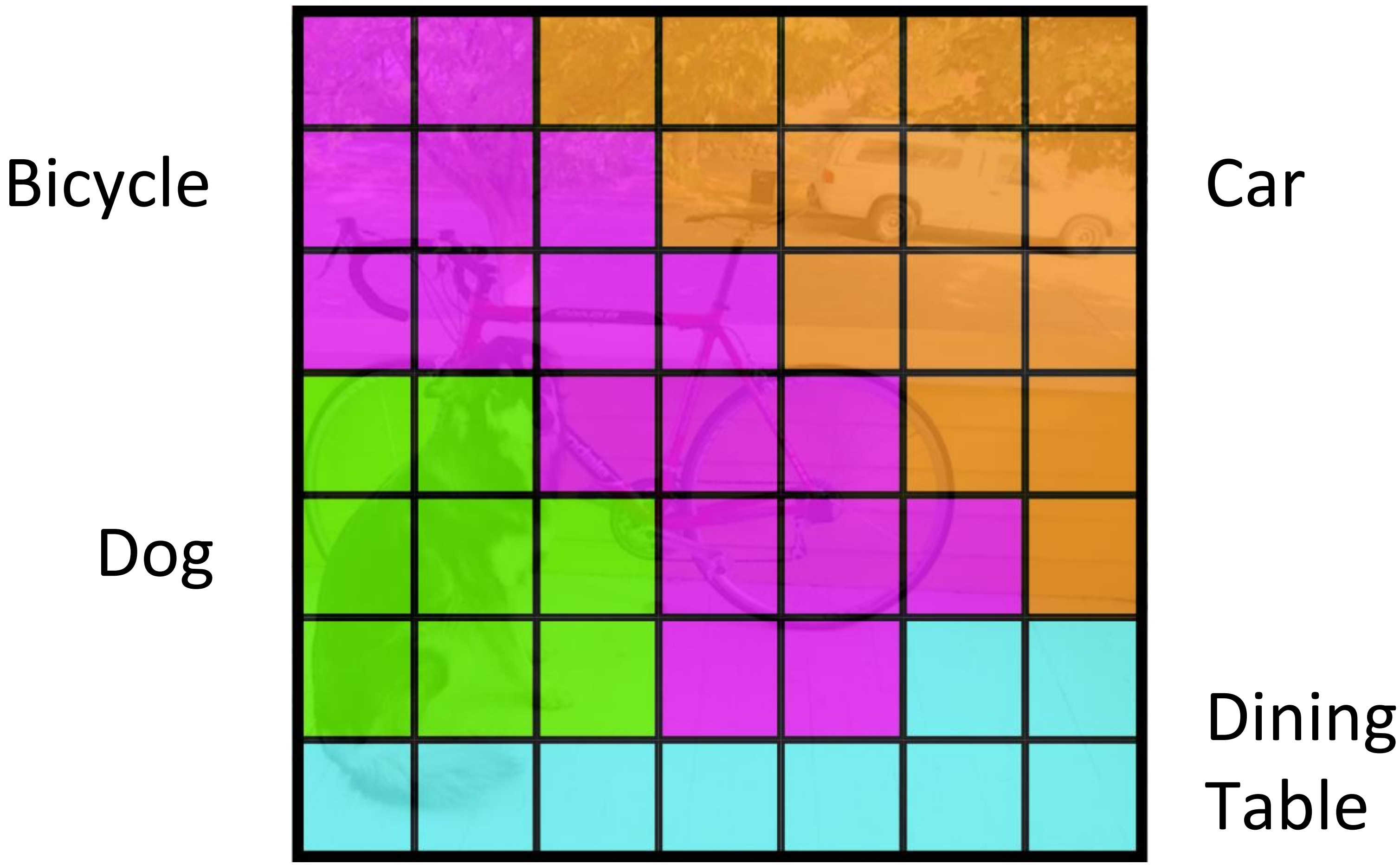
Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Each cell predicts boxes and confidences: P(Object)



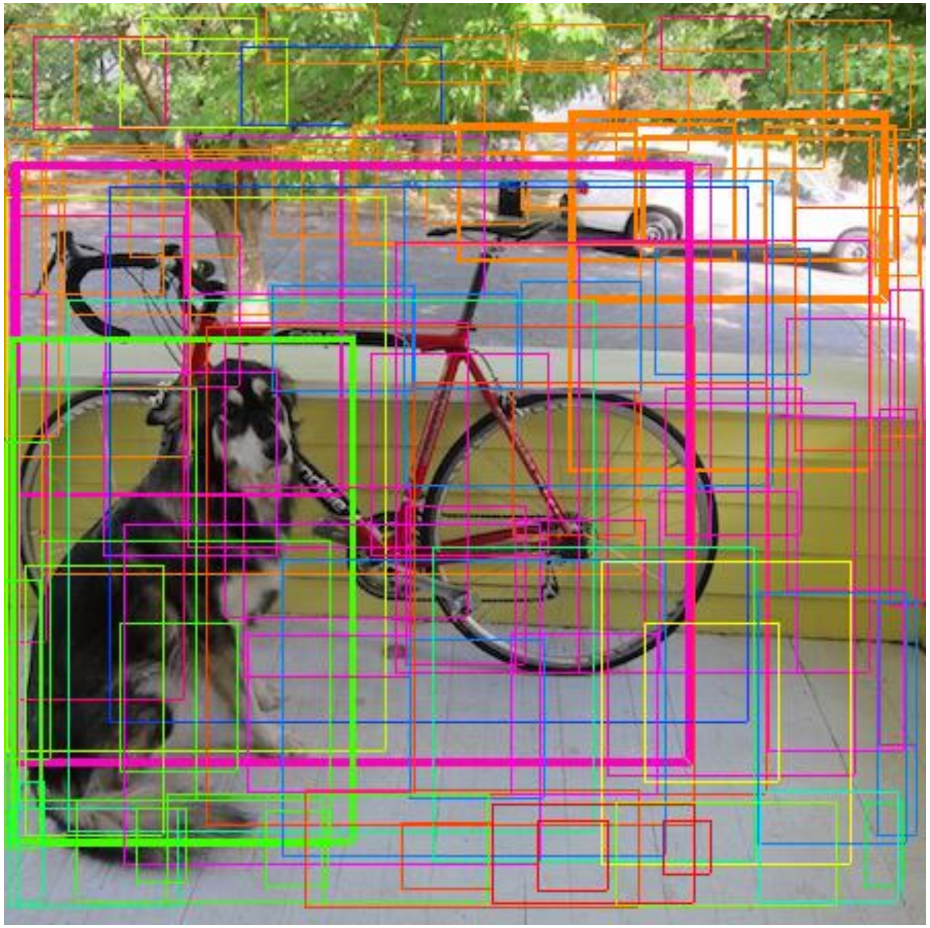
Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Each cell also predicts a probability P(Class | Object)



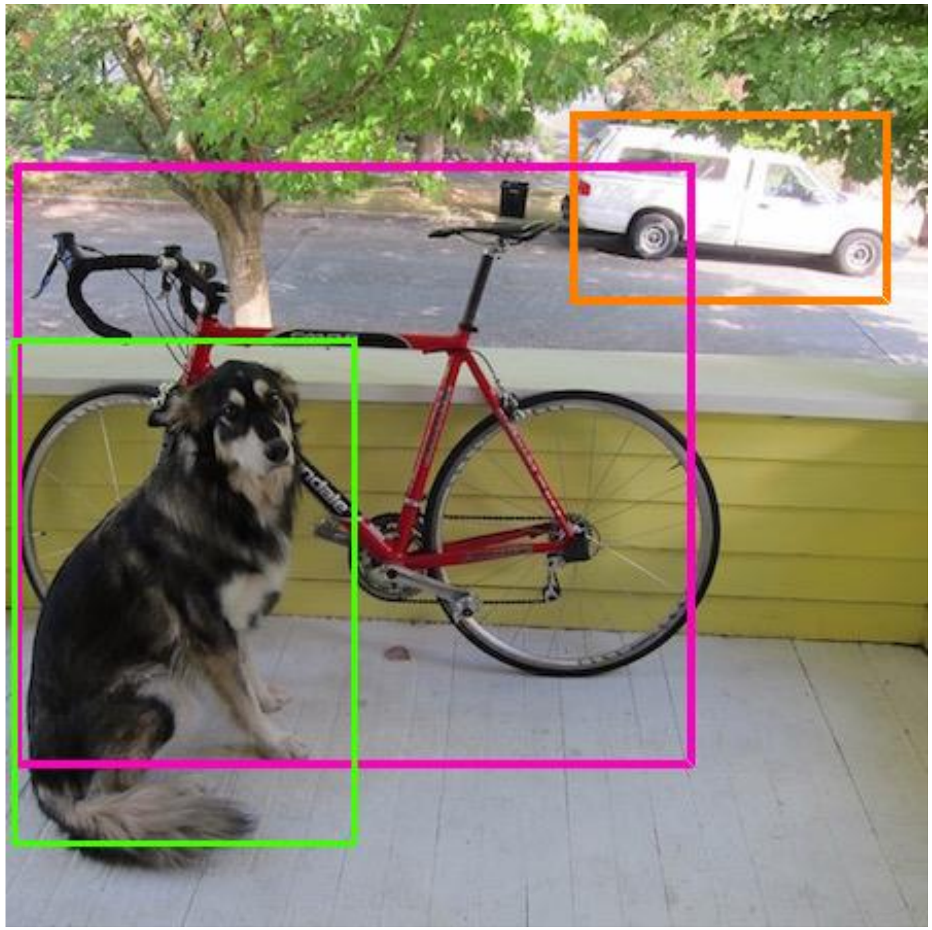
Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Combine the box and class predictions



Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

Finally do NMS and threshold detections

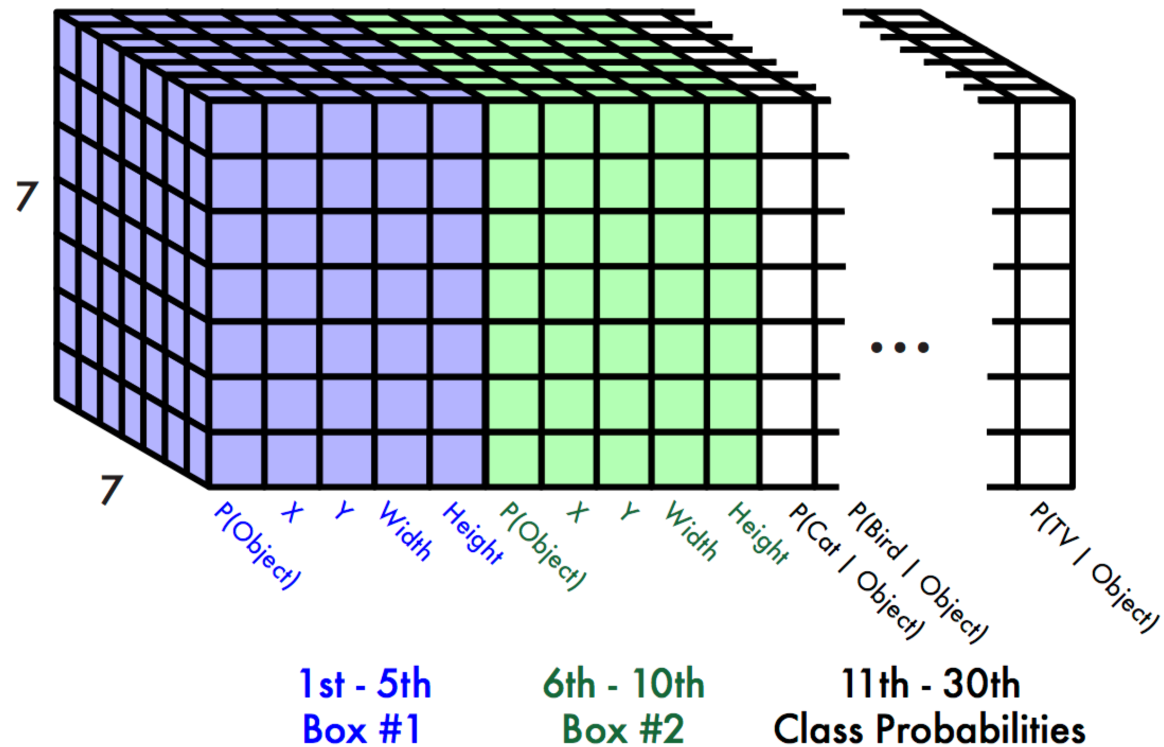


Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

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



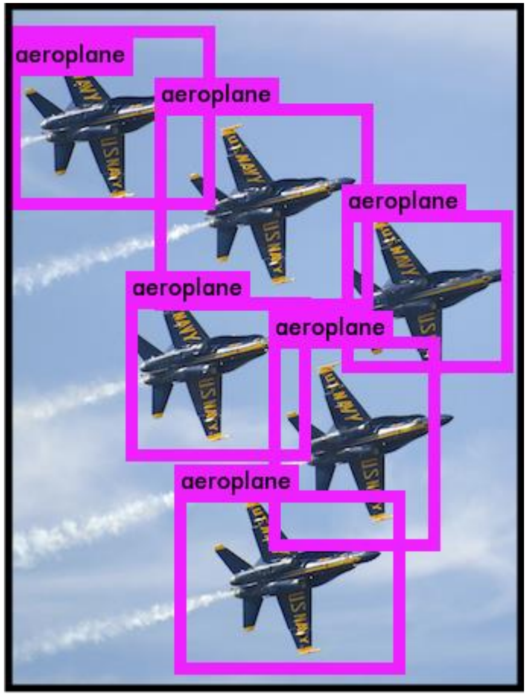
For Pascal VOC:

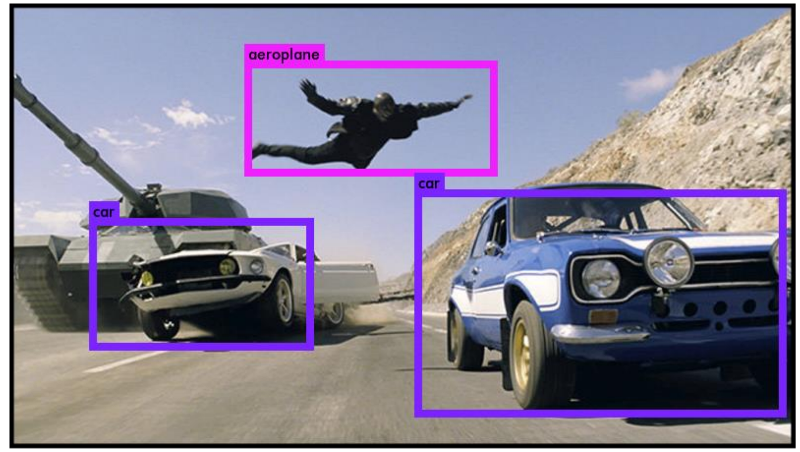
- 7x7 grid
- 2 bounding boxes / cell
- 20 classes

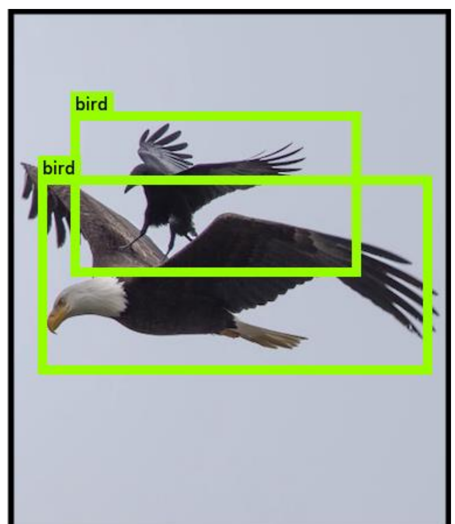
7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

YOLO works across many natural images

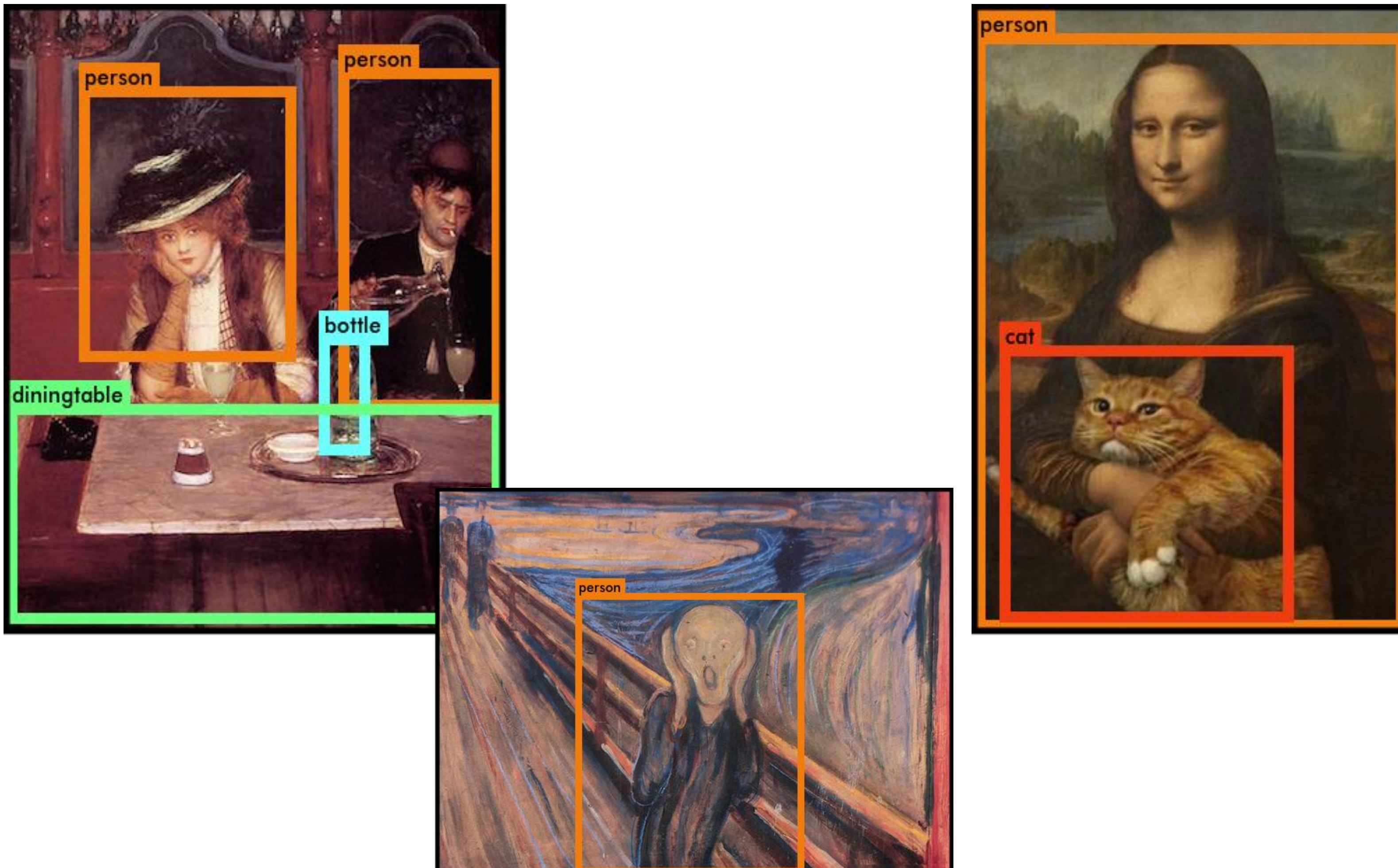






Redmon et al., "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016

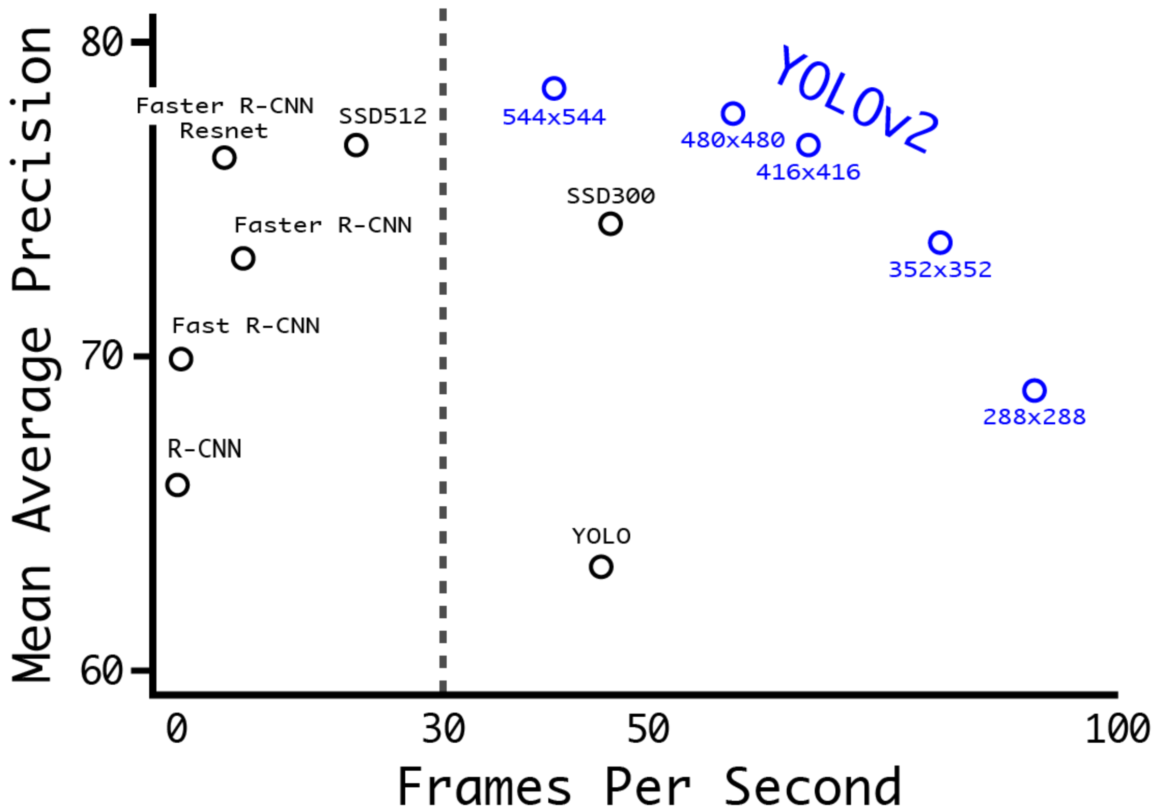
It also generalizes well to new domains



Redmon et al., “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016

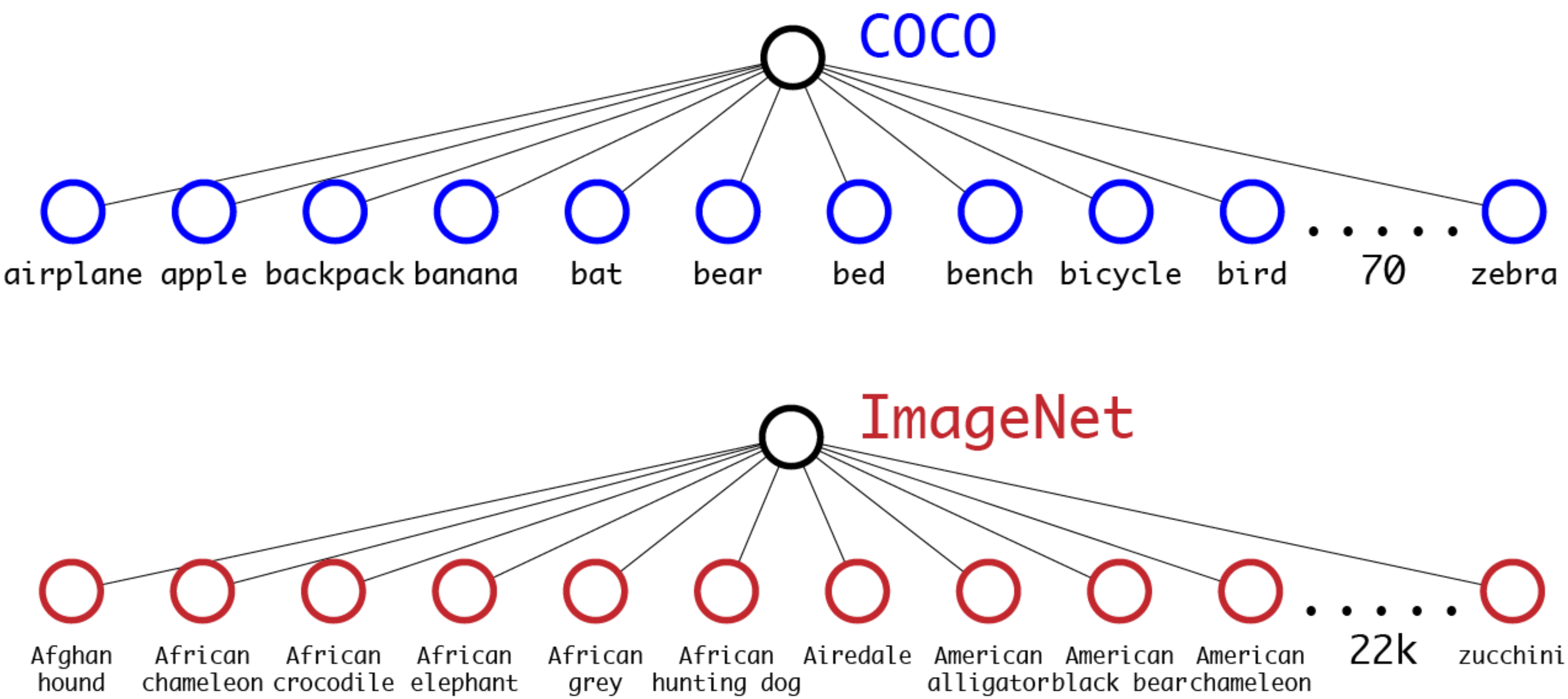
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

YOLOv2: Fast, Accurate Detection



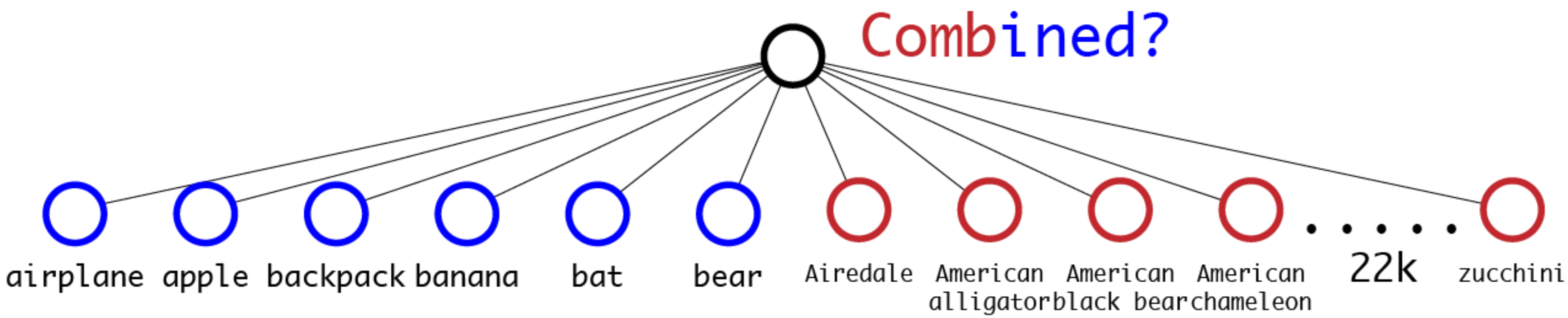
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

Typically use softmax over all classes



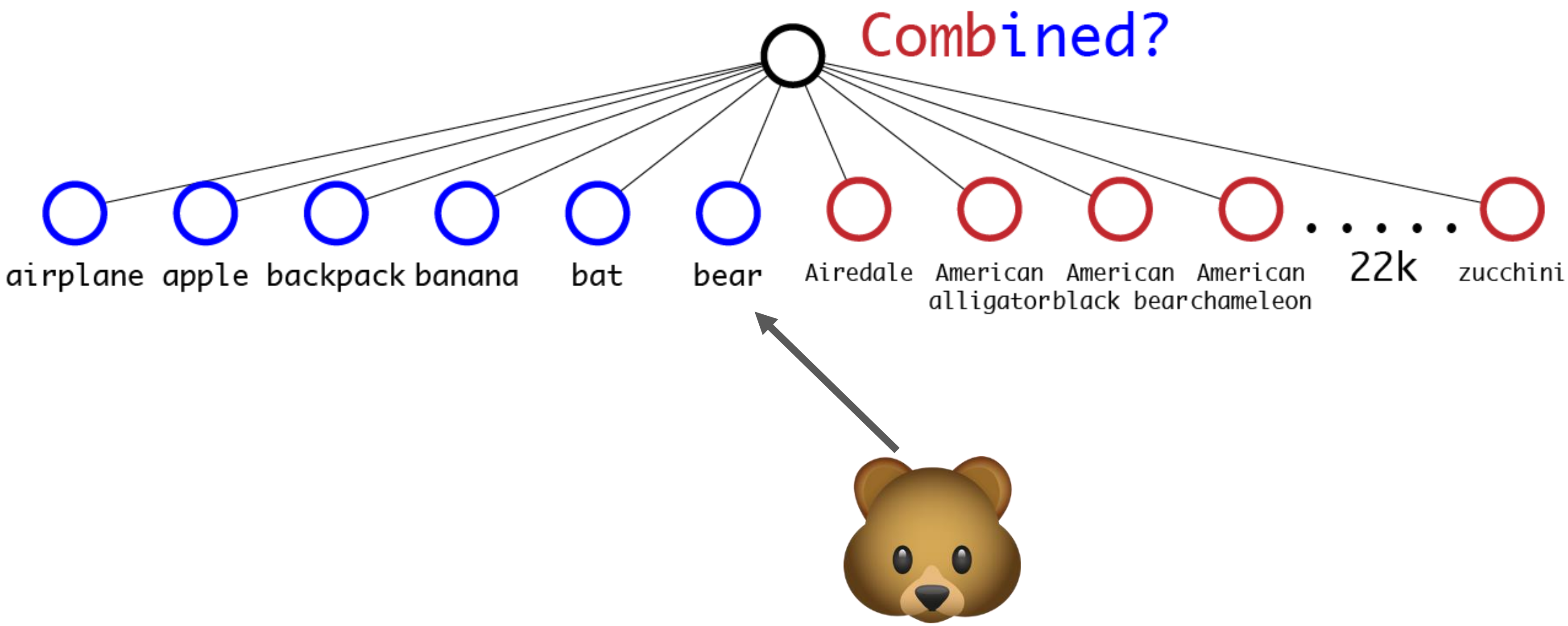
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

Can’t just mash classes together...



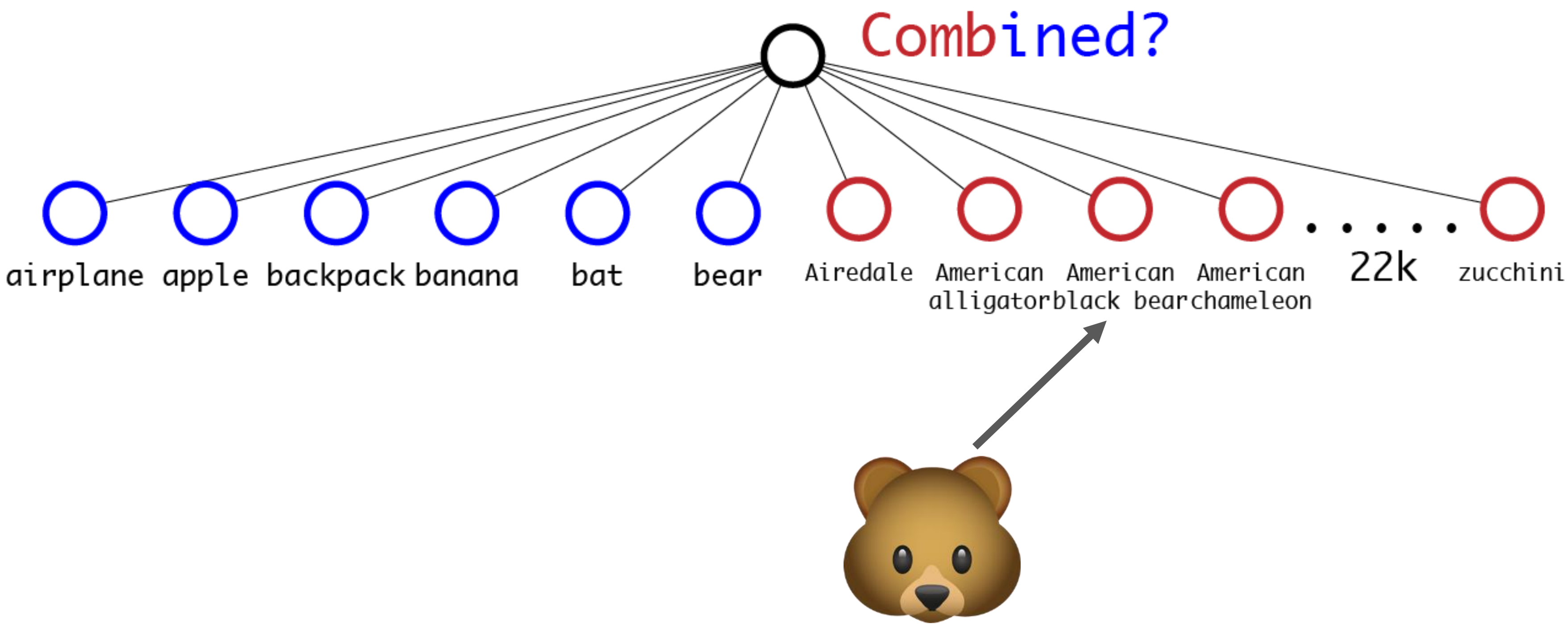
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017

Can't just mash classes together...



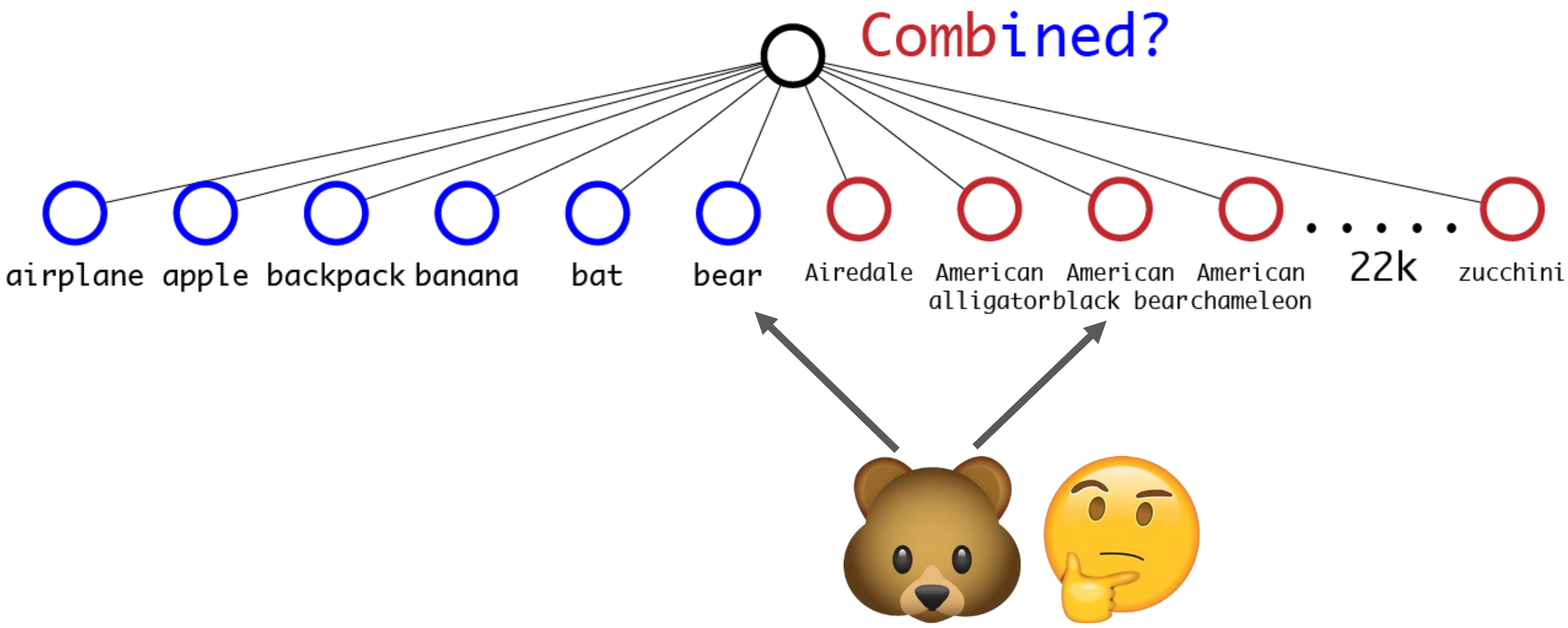
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

Can't just mash classes together...

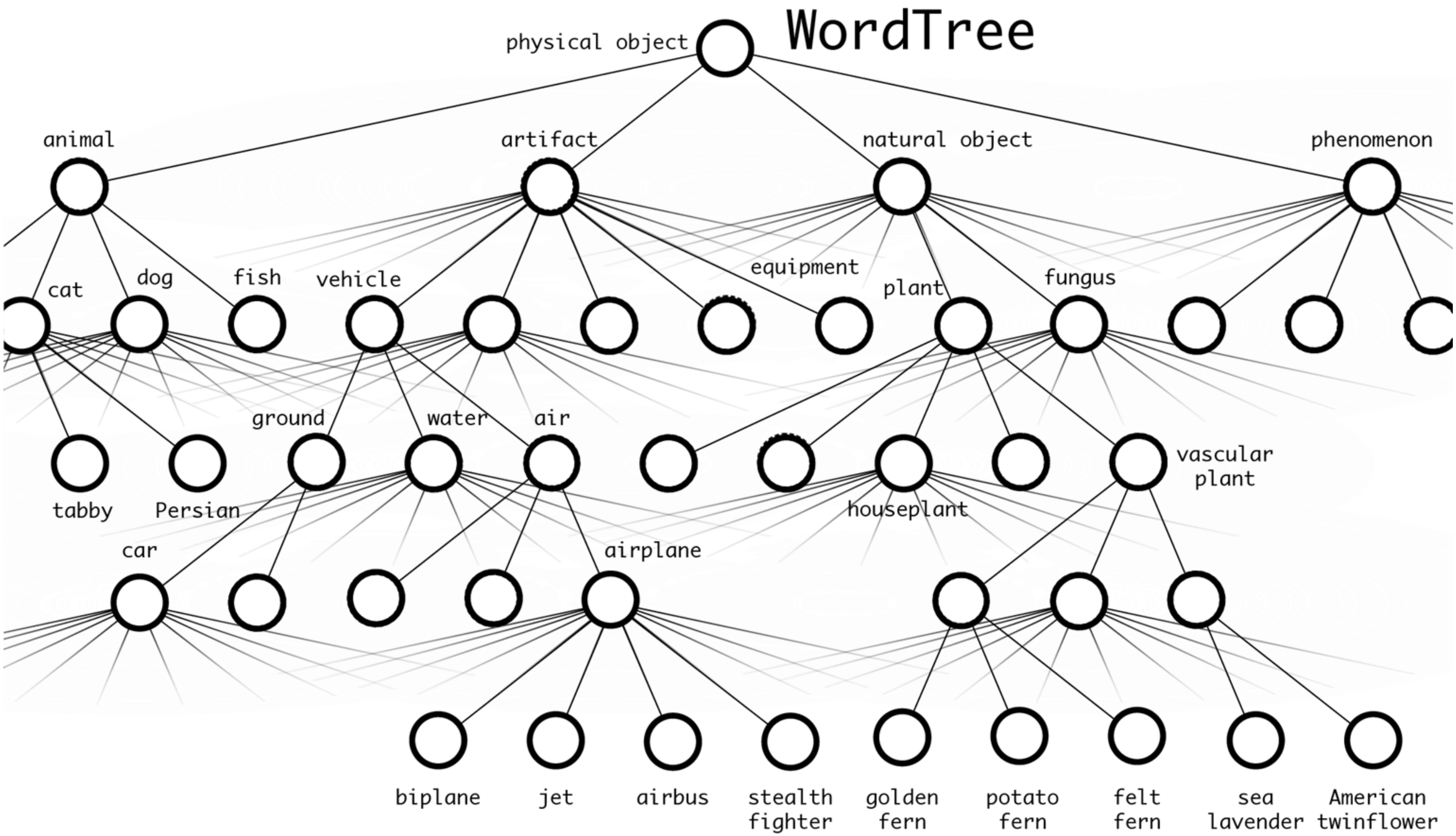


Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

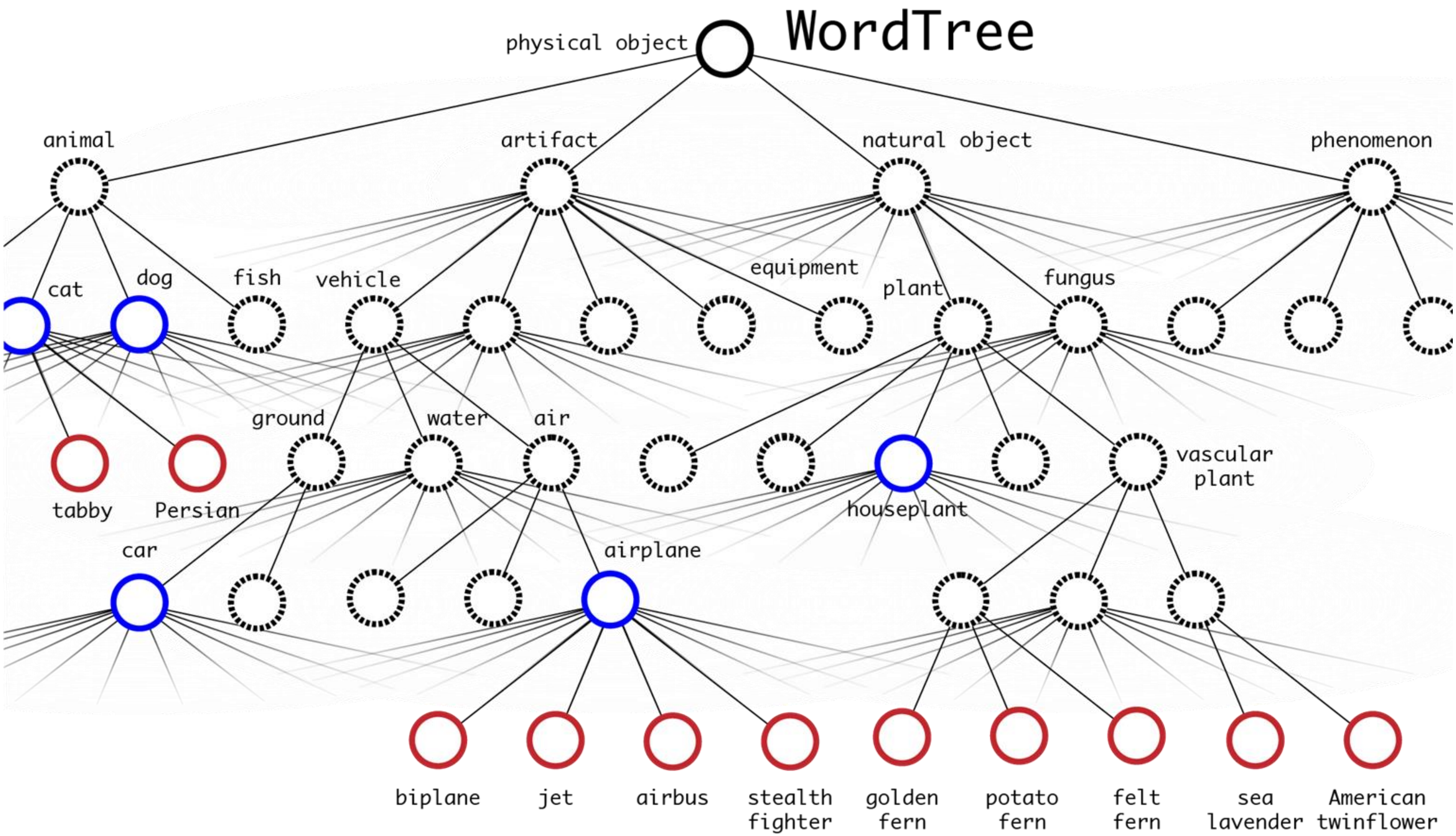
Can't just mash classes together...



Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

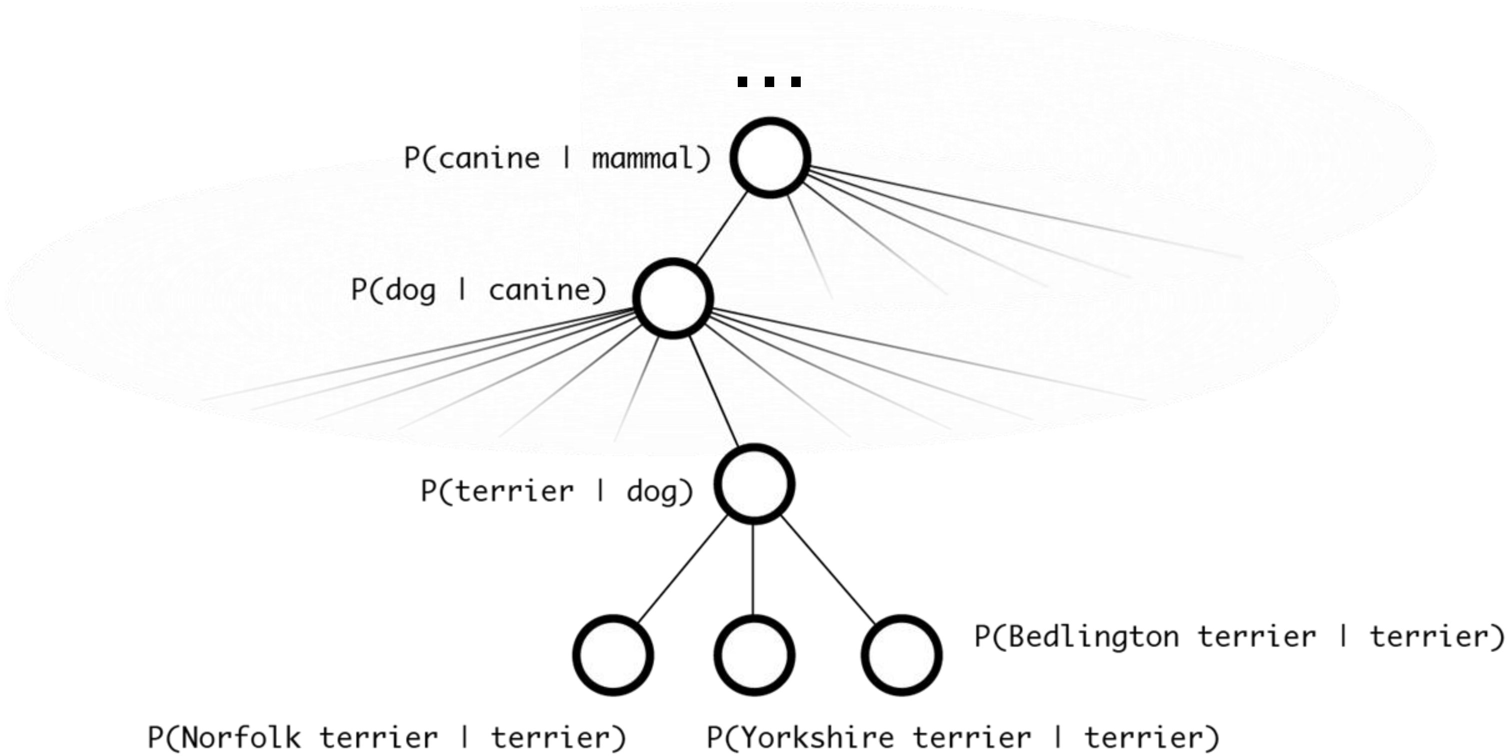


Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017



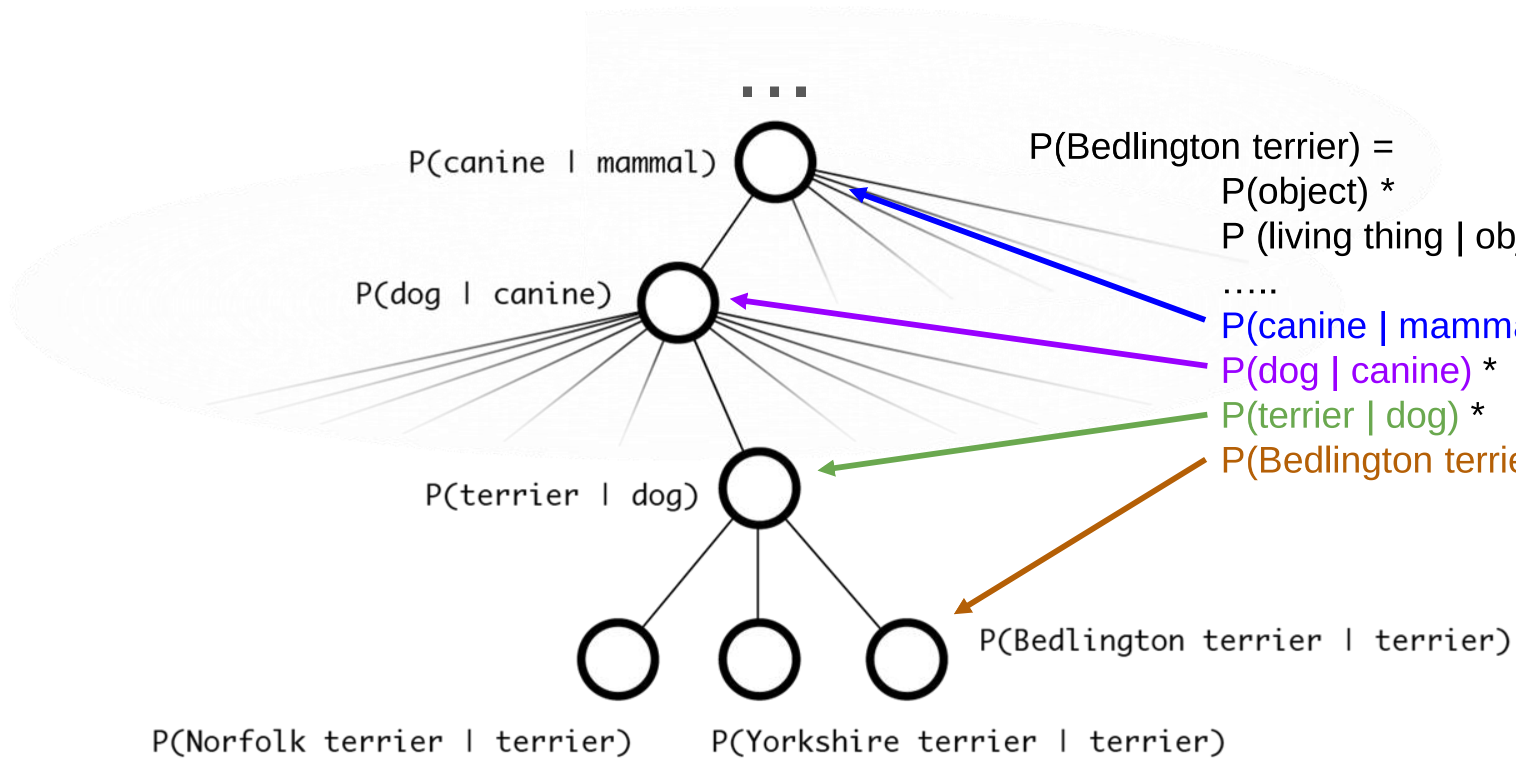
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

Each node is a conditional probability



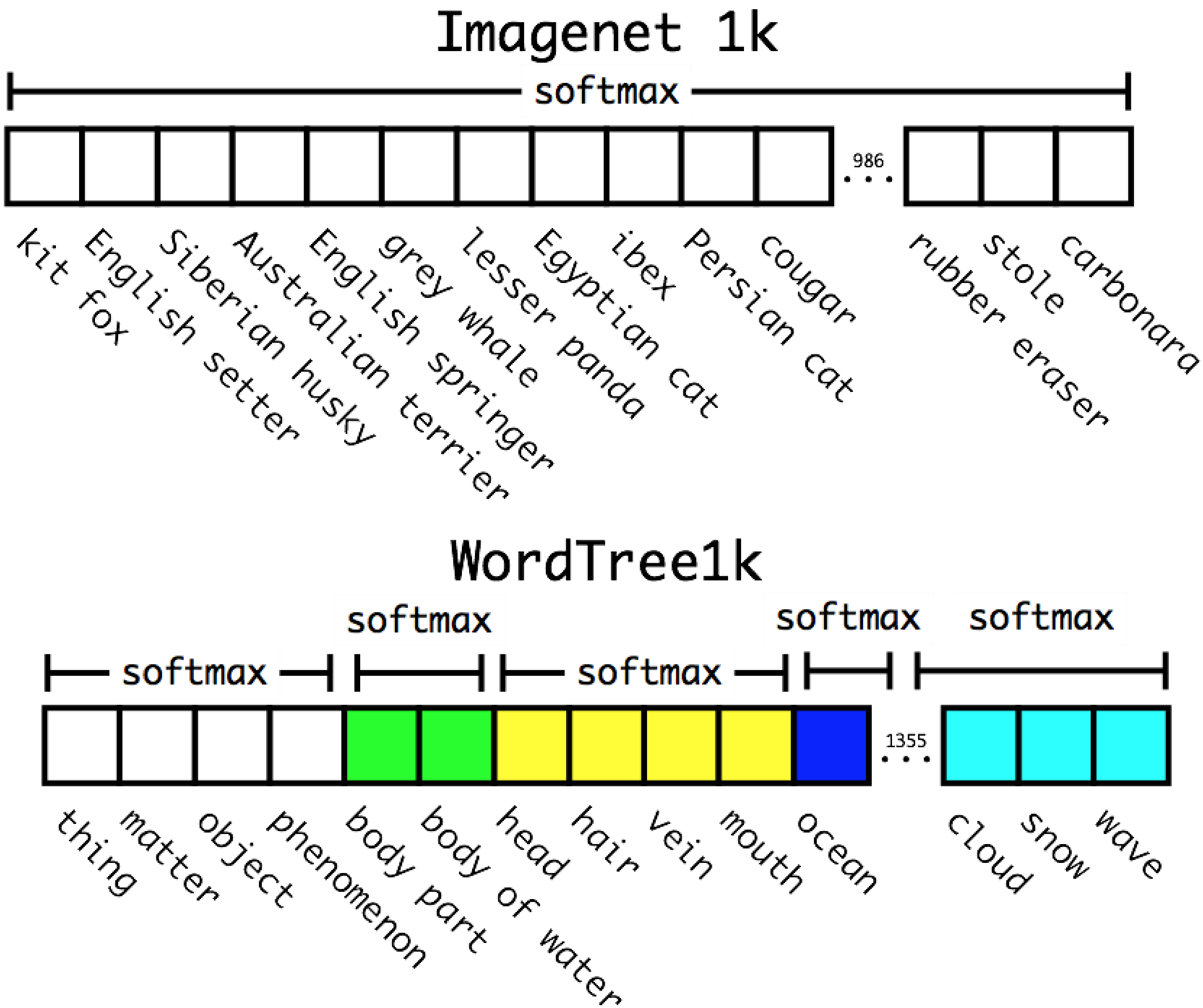
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

Each node is a conditional probability

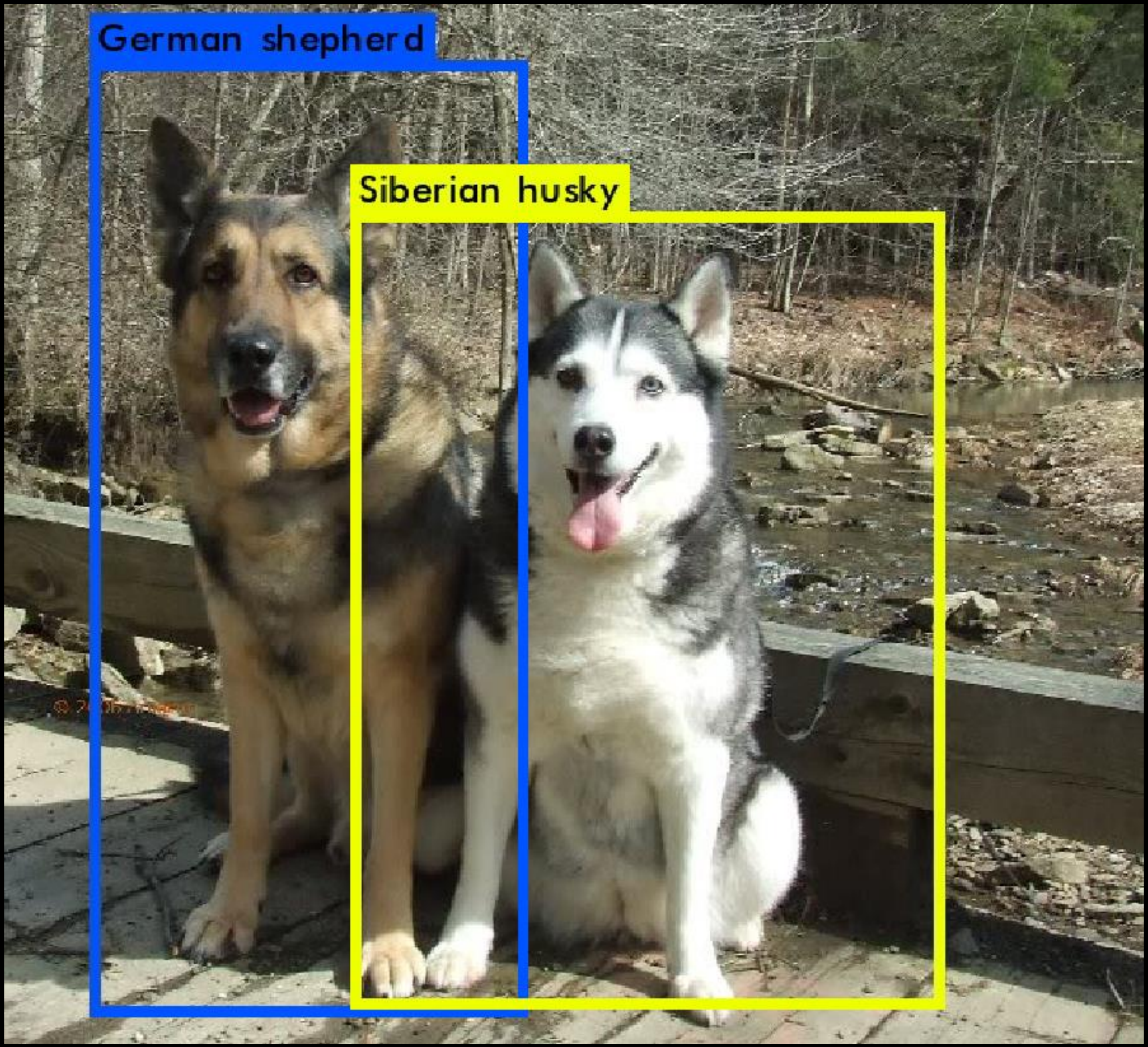


P(Bedlington terrier) =
P(object) *
P (living thing | object) *
.....
P(canine | mammal) *
P(dog | canine) *
P(terrier | dog) *
P(Bedlington terrier | terrier)

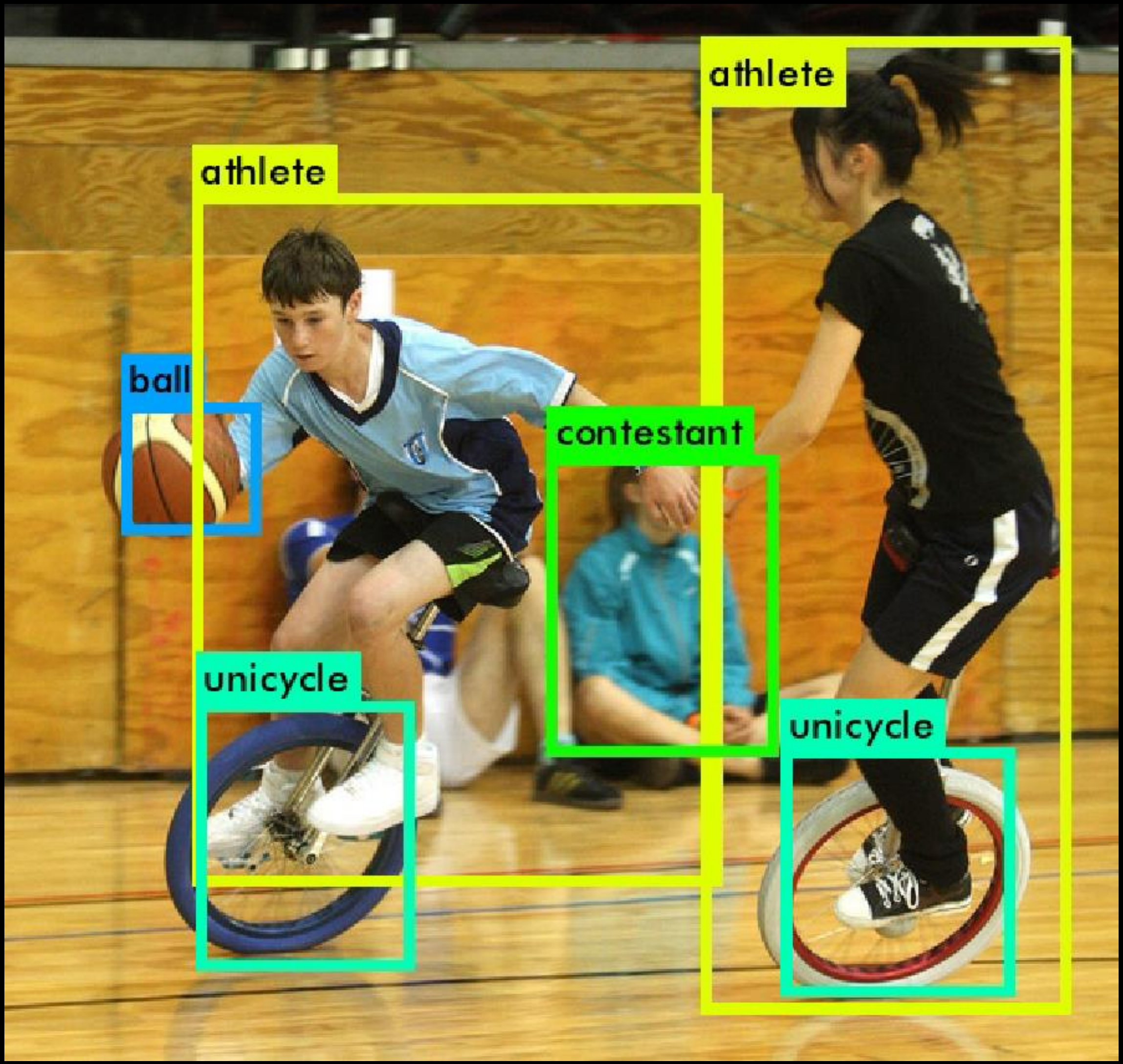
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017



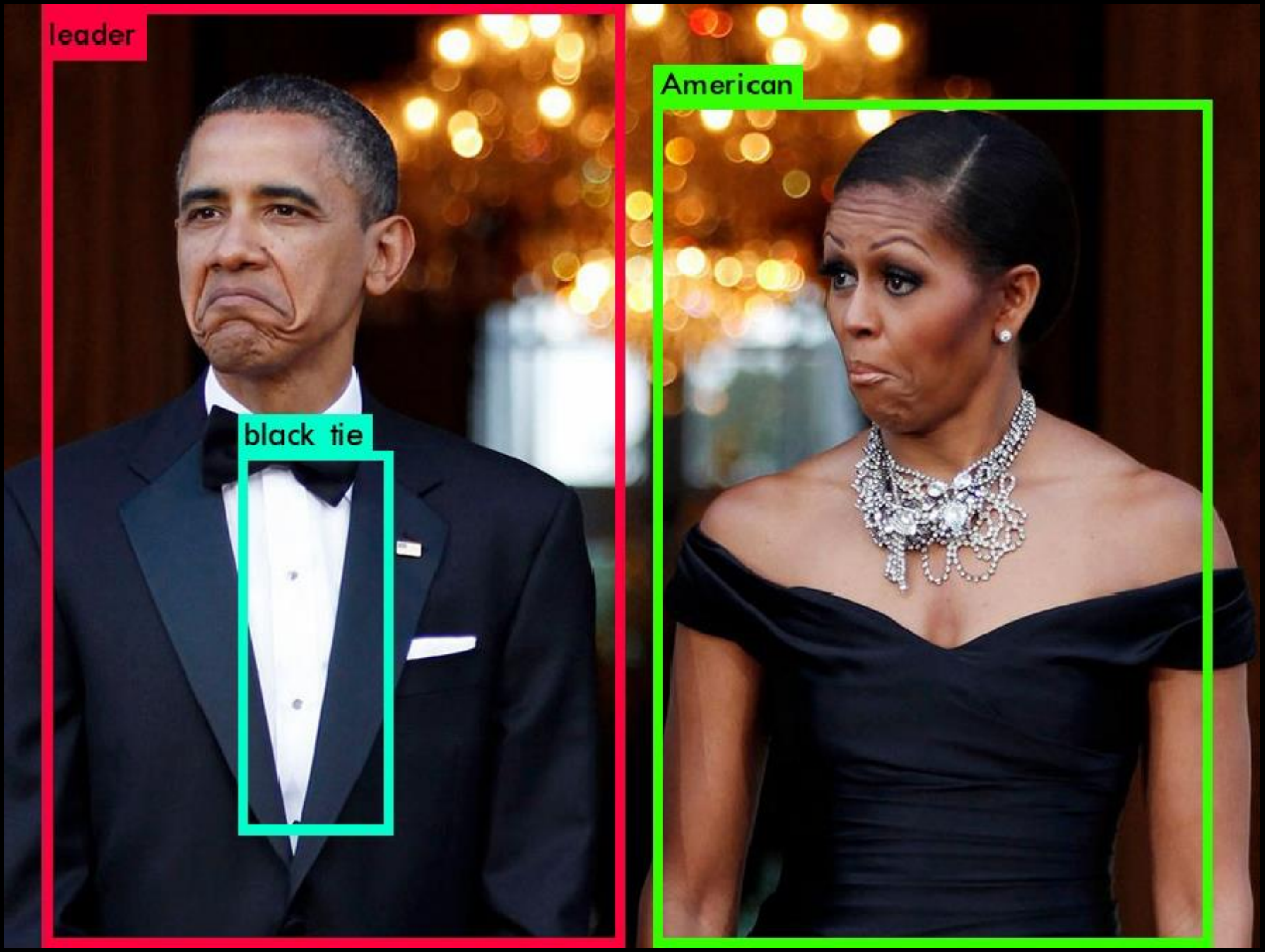
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017



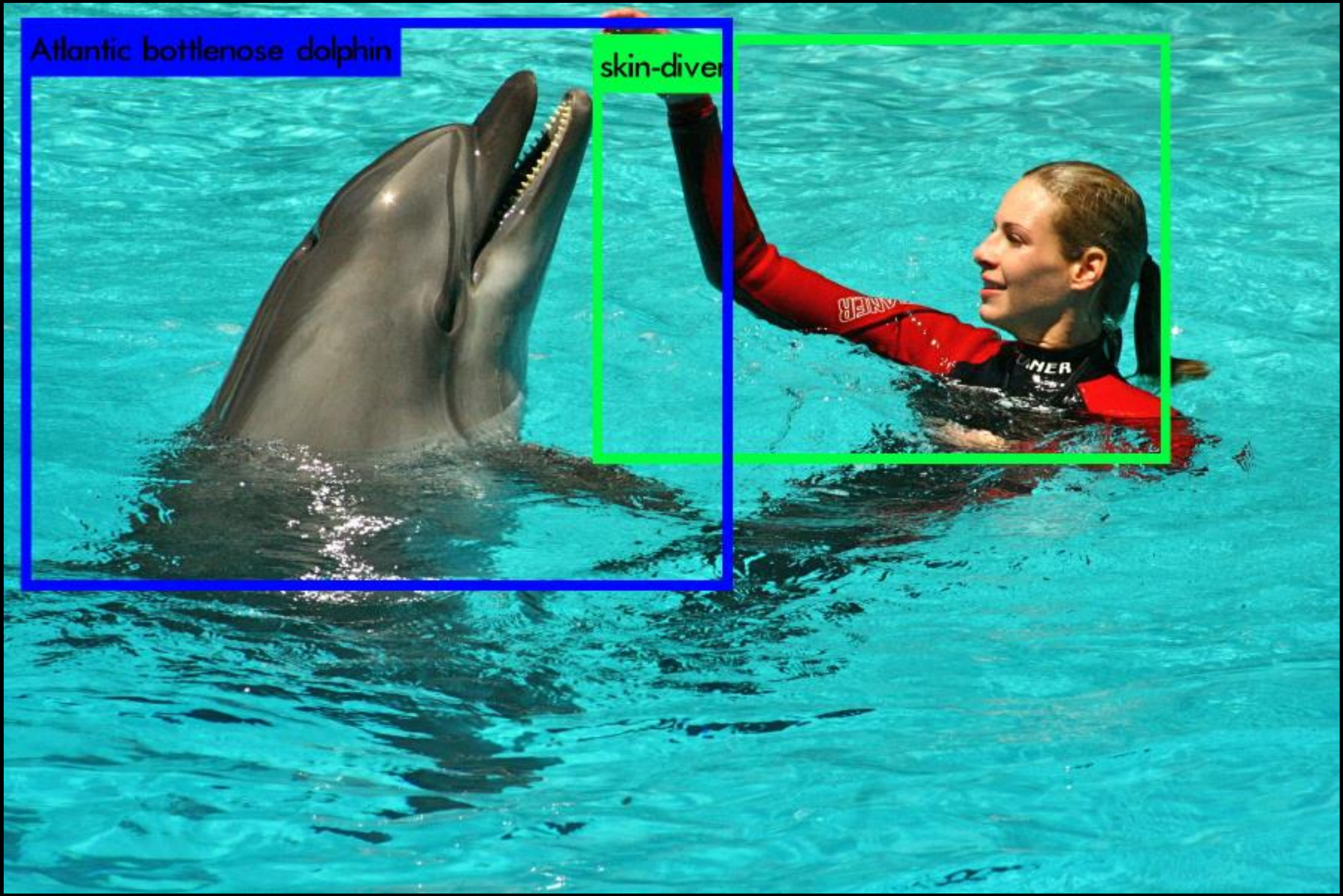
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017



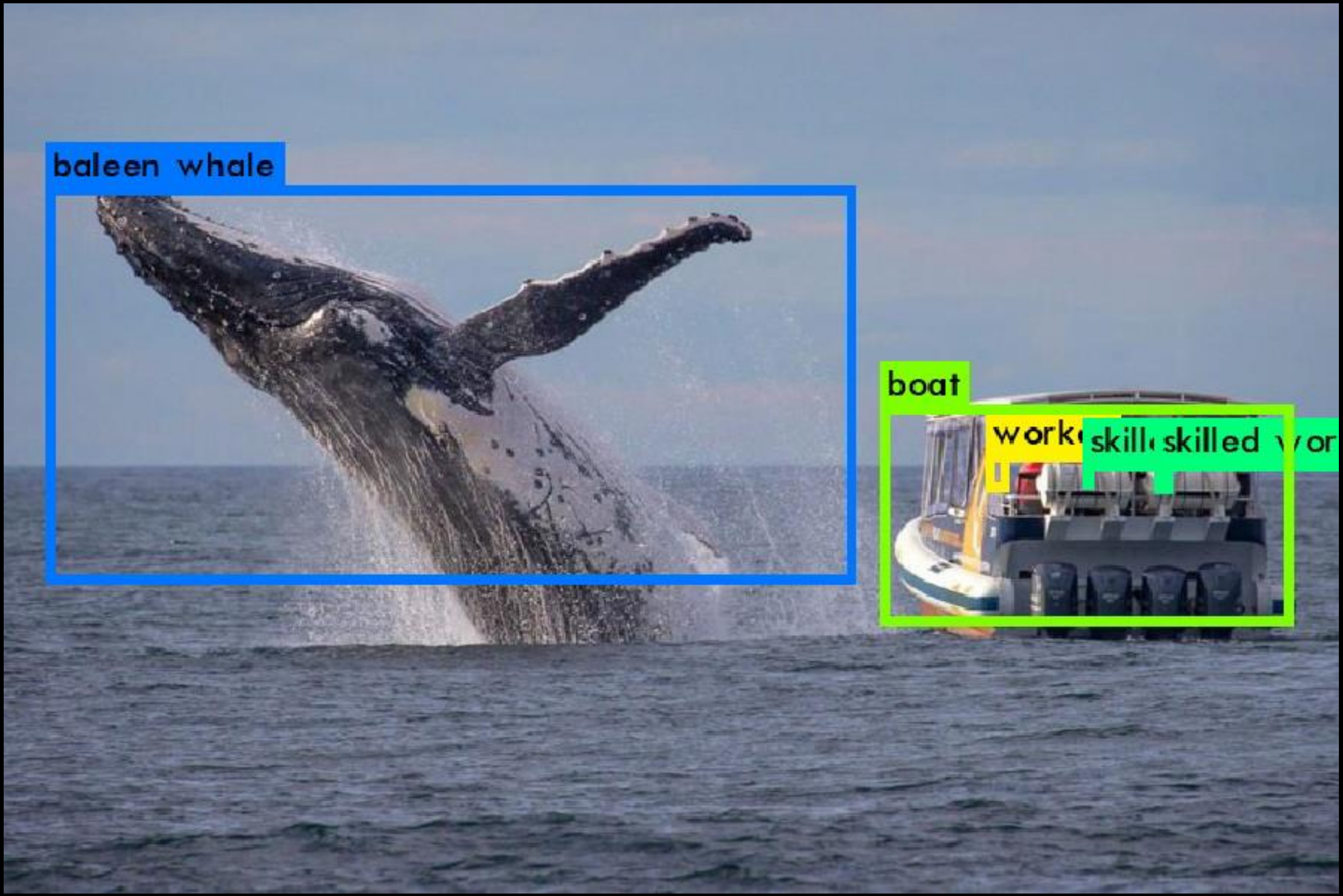
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017



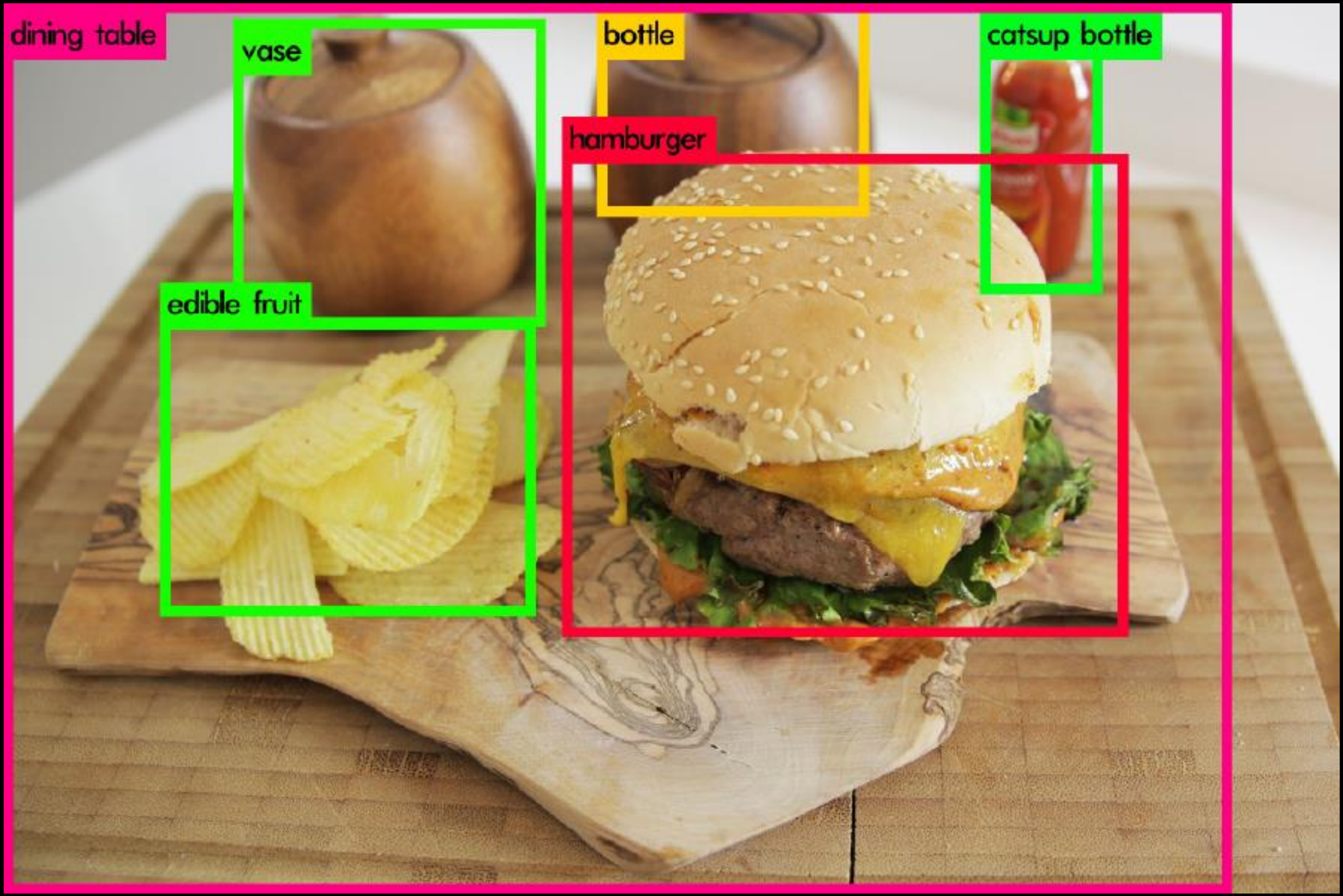
Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017



Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017



Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017



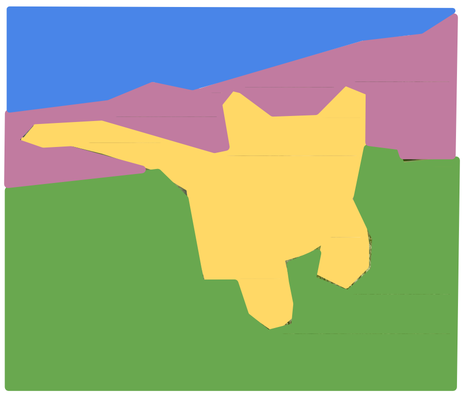
Redmon and Farhadi, “YOLO9000: Better, Faster, Stronger”, CVPR 2017

Redmon and Farhadi, "YOLO9000: Better, Faster, Stronger", CVPR 2017

Plan for this lecture

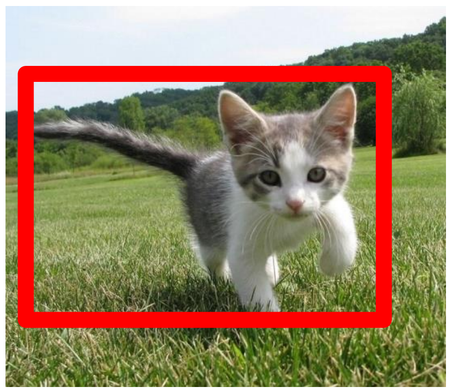
- Fully supervised detection
  - Pre-CNN: Deformable part models
  - Detection with region proposals: R-CNN, Fast/er R-CNN
  - Detection without region proposals: YOLO
  - Semantic and instance segmentation: FCN, Mask R-CNN
- Weak or out-of-domain supervision
  - Weakly supervised object detection
  - Domain adaptation

Semantic Segmentation



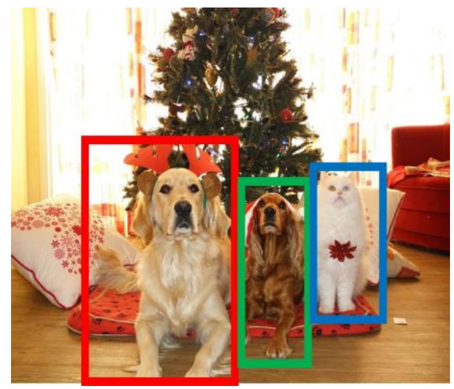
GRASS, CAT, TREE, SKY

No objects, just pixels



CAT

Single Object



DOG, DOG, CAT

DOG, DOG, CAT

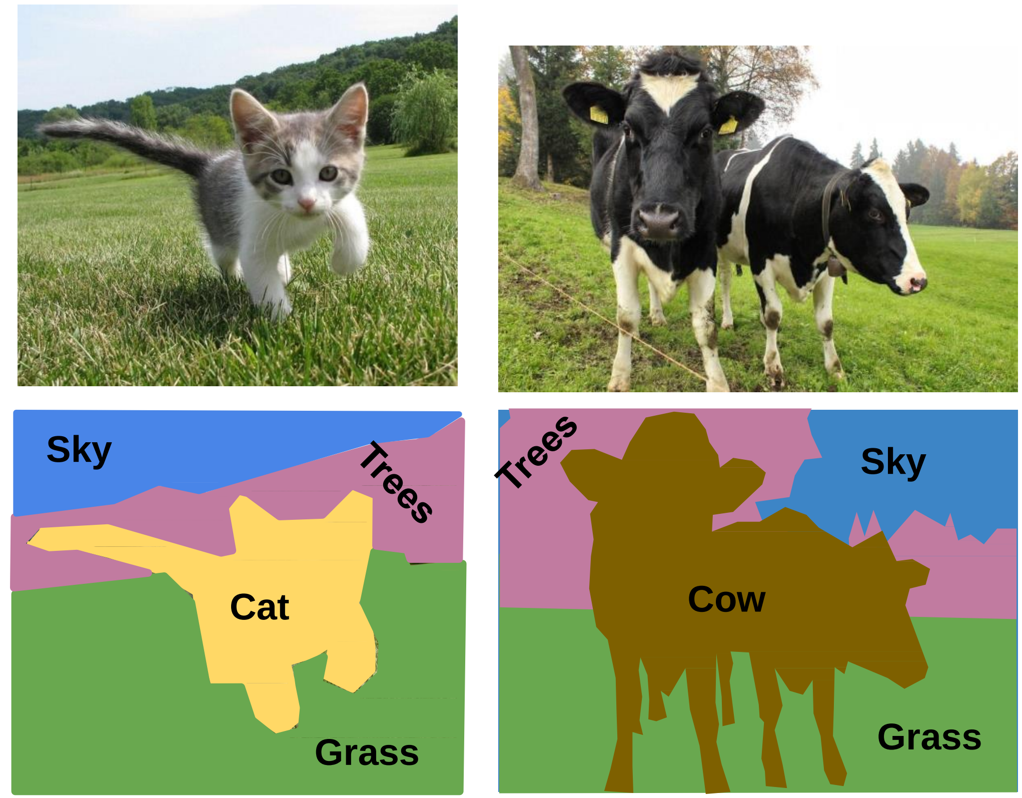
Multiple Object

Slide by: Justin Johnson

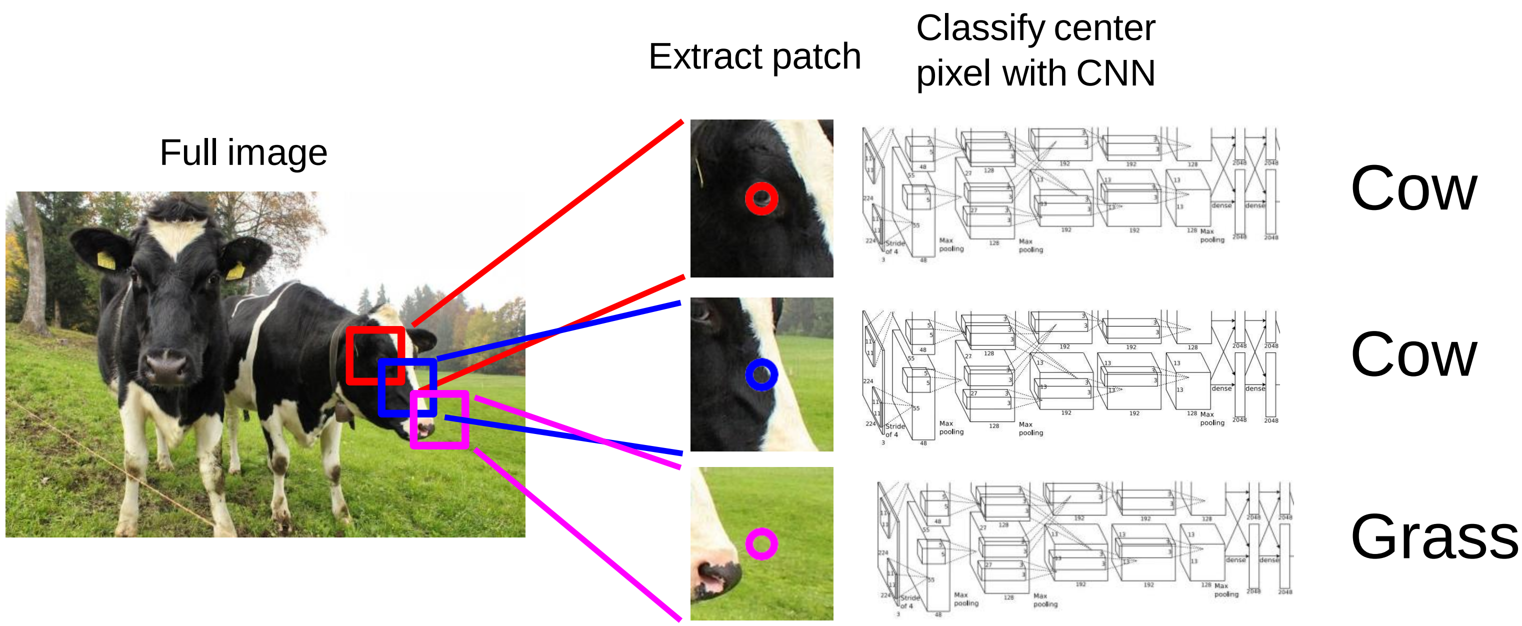
Semantic Segmentation

Label each pixel in the image with a category label

Don’t differentiate instances, only care about pixels



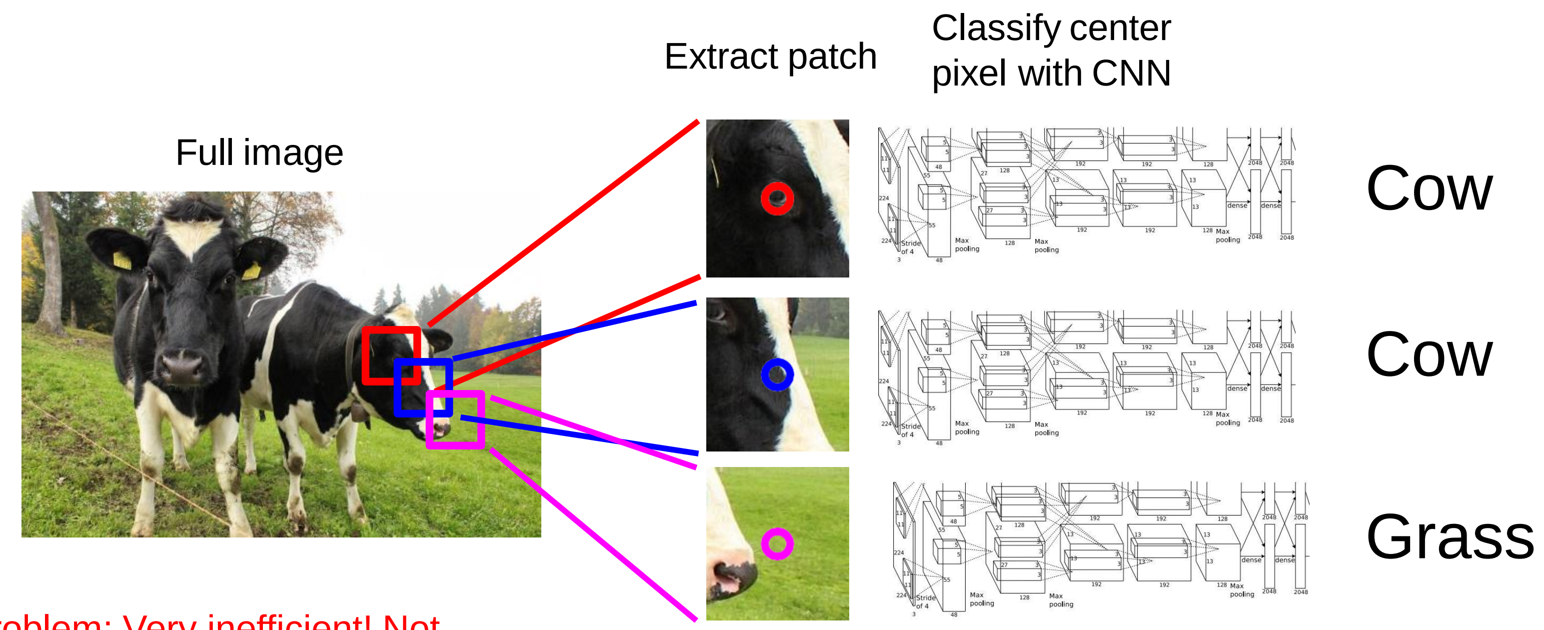
Semantic Segmentation Idea: Sliding Window



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

Slide by: Justin Johnson

Semantic Segmentation Idea: Sliding Window



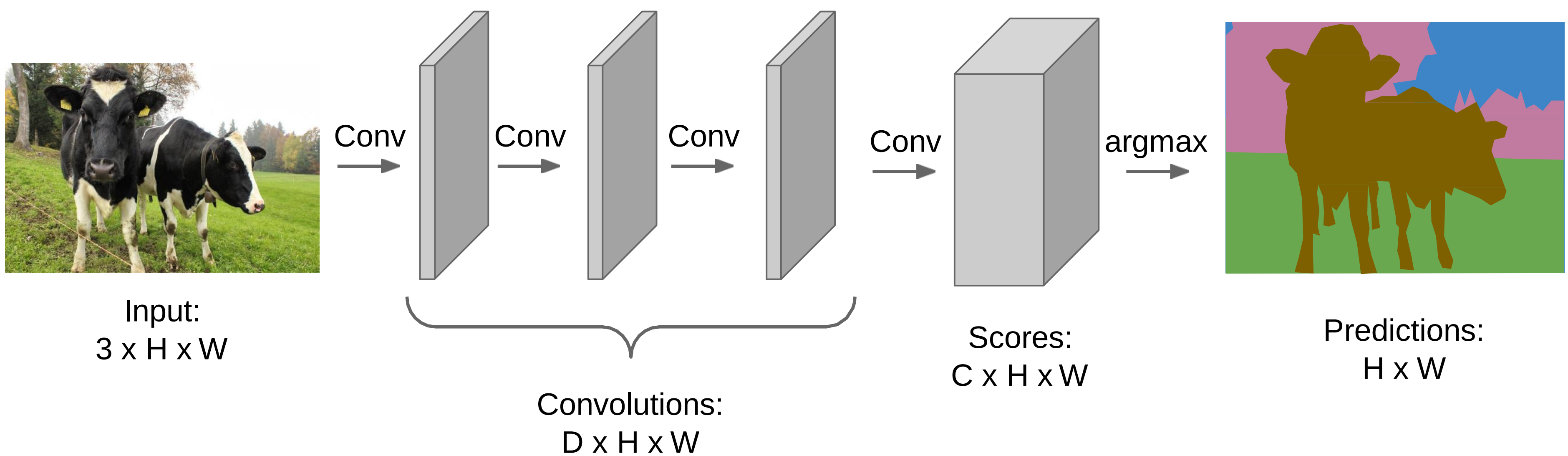
Problem: Very inefficient! Not reusing shared features between overlapping patches

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

Slide by: Justin Johnson

Semantic Segmentation Idea:
Fully Convolutional

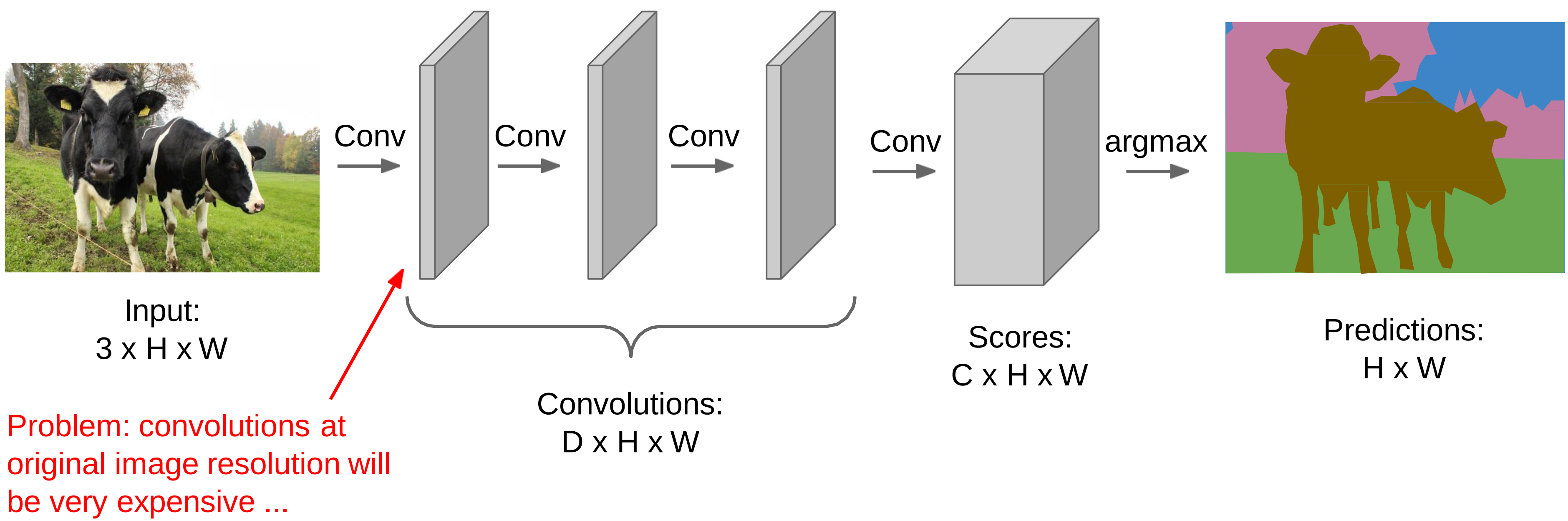
Design a network as a bunch of convolutional layers
to make predictions for pixels all at once!



Slide by: Justin Johnson

Semantic Segmentation Idea: Fully Convolutional

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

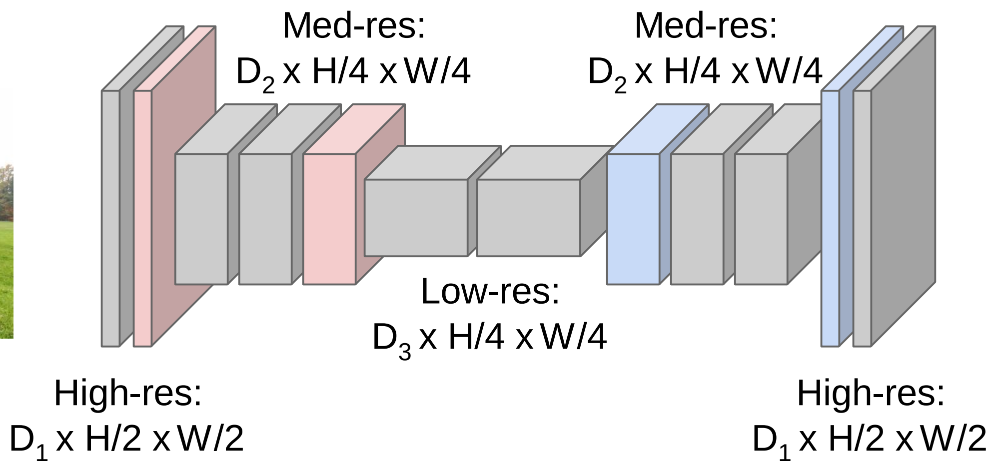


Slide by: Justin Johnson

Semantic Segmentation Idea: Fully Convolutional

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

Input:
3 x H x W



Predictions:
H x 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 by: Justin Johnson

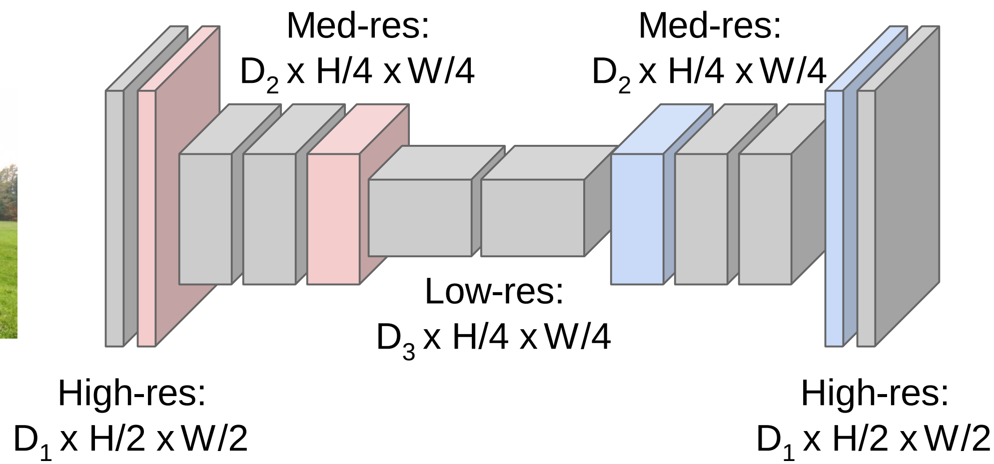
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: ???

Input:
3 x H x W



Predictions:
H x 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 by: Justin Johnson

In-Network upsampling: “Unpooling”

Nearest Neighbor

1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

“Bed of Nails”

1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

Slide by: Justin Johnson

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

…

Rest of the network

Max Unpooling
Use positions from pooling layer

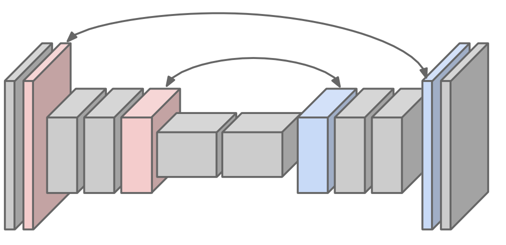
1	2
3	4

Input: 2 x 2

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

Output: 4 x 4

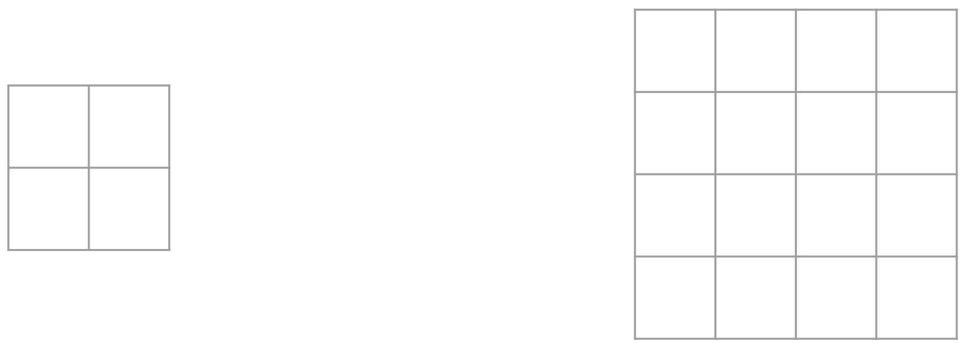
Corresponding pairs of downsampling and upsampling layers



Slide by: Justin Johnson

Learnable Upsampling: Transpose Convolution

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



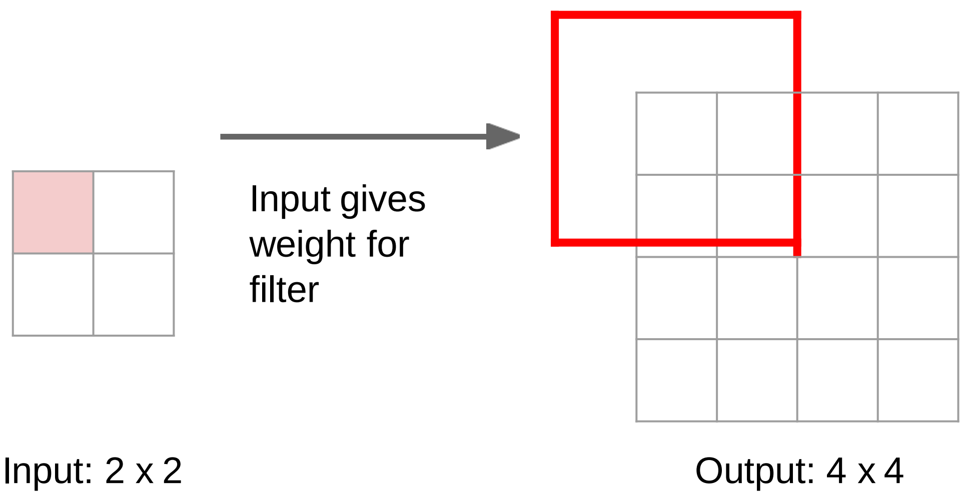
Input: 2 x 2

Output: 4 x 4

Slide by: Justin Johnson

Learnable Upsampling: Transpose Convolution

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

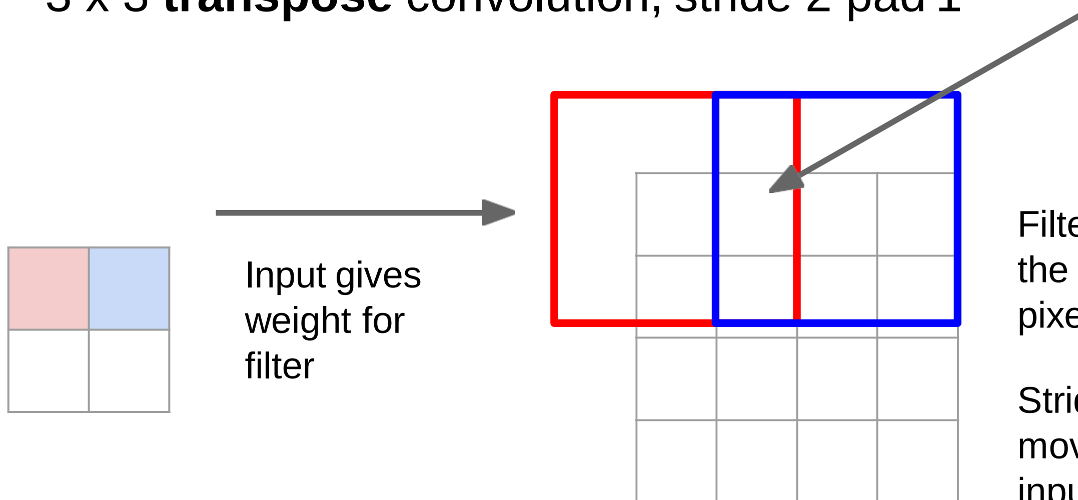


Slide by: Justin Johnson

Learnable Upsampling: Transpose Convolution

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

Sum where output overlaps



Input: 2 x 2

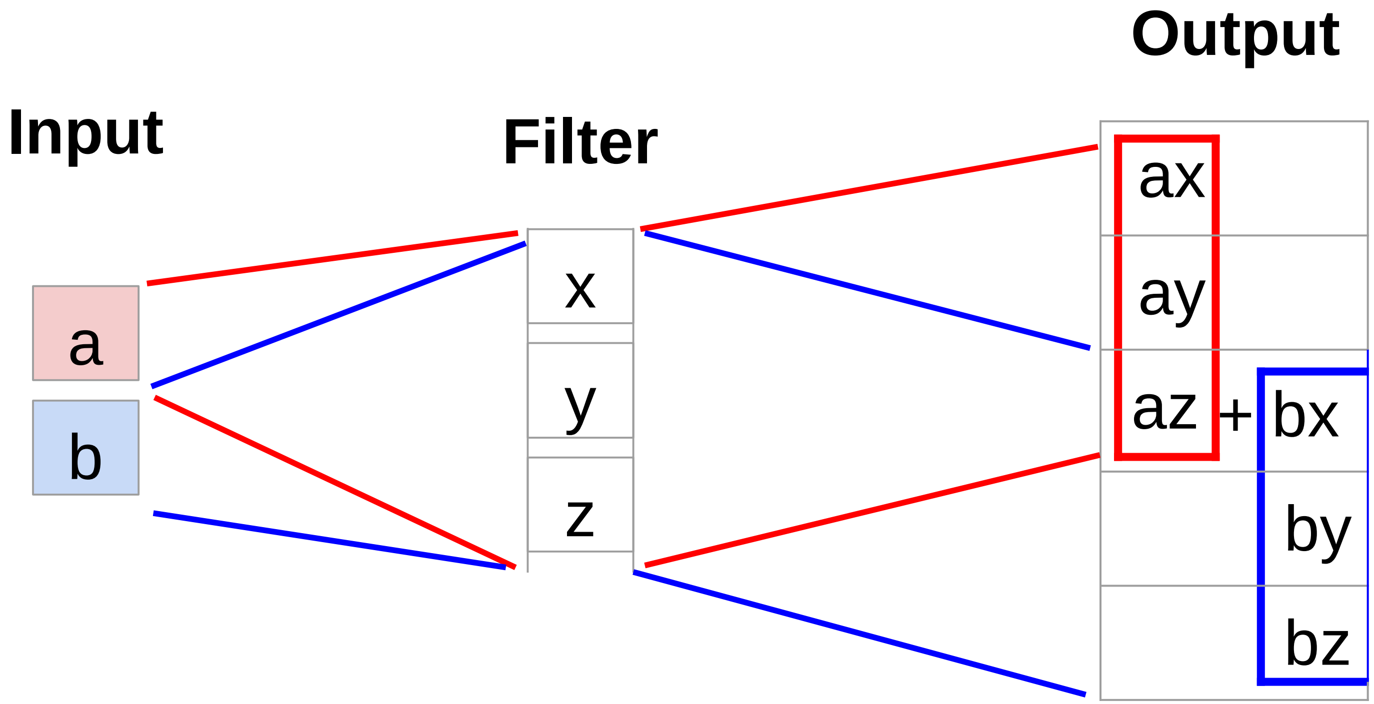
Output: 4 x 4

Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

Stride gives ratio between movement in output and input

Slide by: Justin Johnson

Transpose Convolution: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Adapted from Justin Johnson

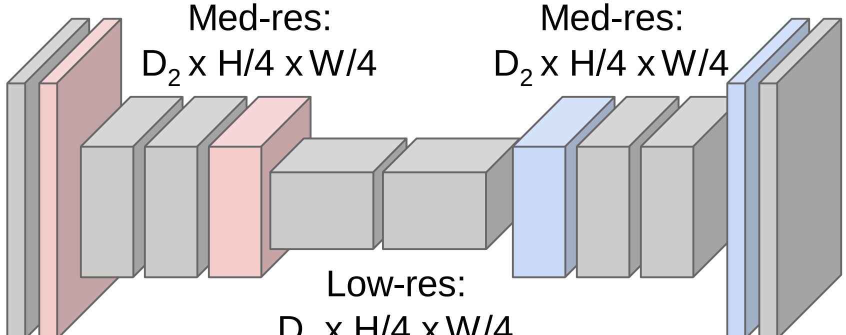
Semantic Segmentation Idea: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: Unpooling or strided transpose convolution

Input: $3 \times H \times W$



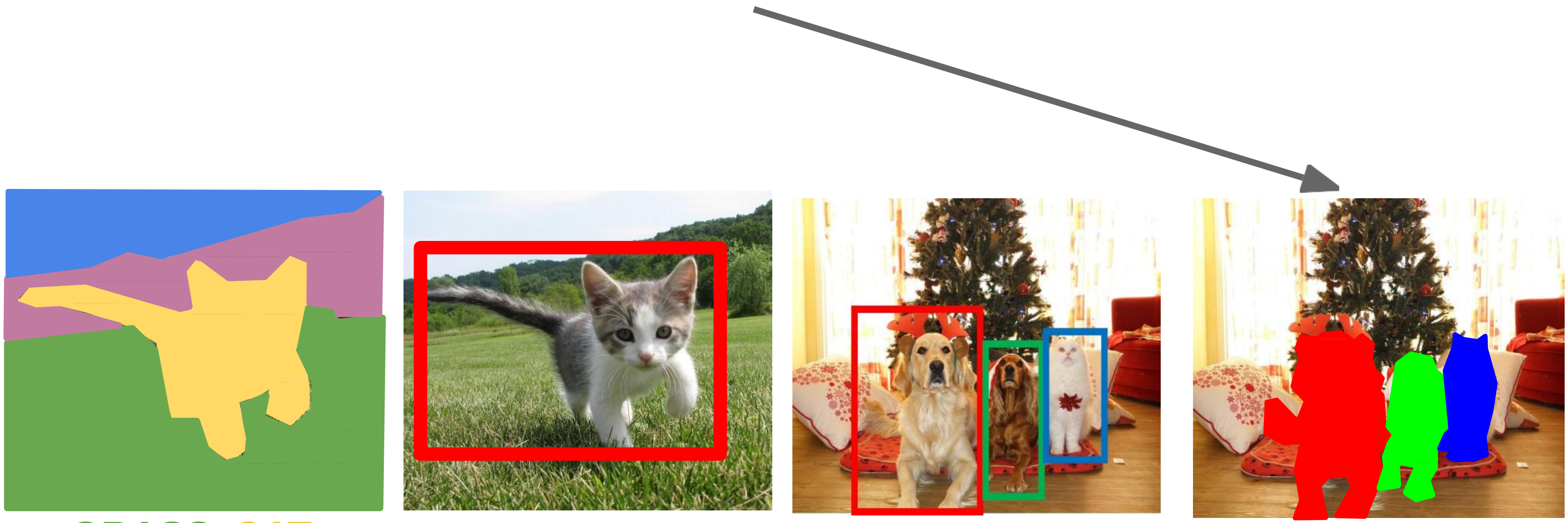
Predictions: $H \times W$

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Slide by: Justin Johnson

Instance Segmentation



GRASS, CAT, TREE, SKY

CAT

DOG, DOG, CAT

DOG, DOG, CAT

No objects, just pixels

Single Object

Multiple Object

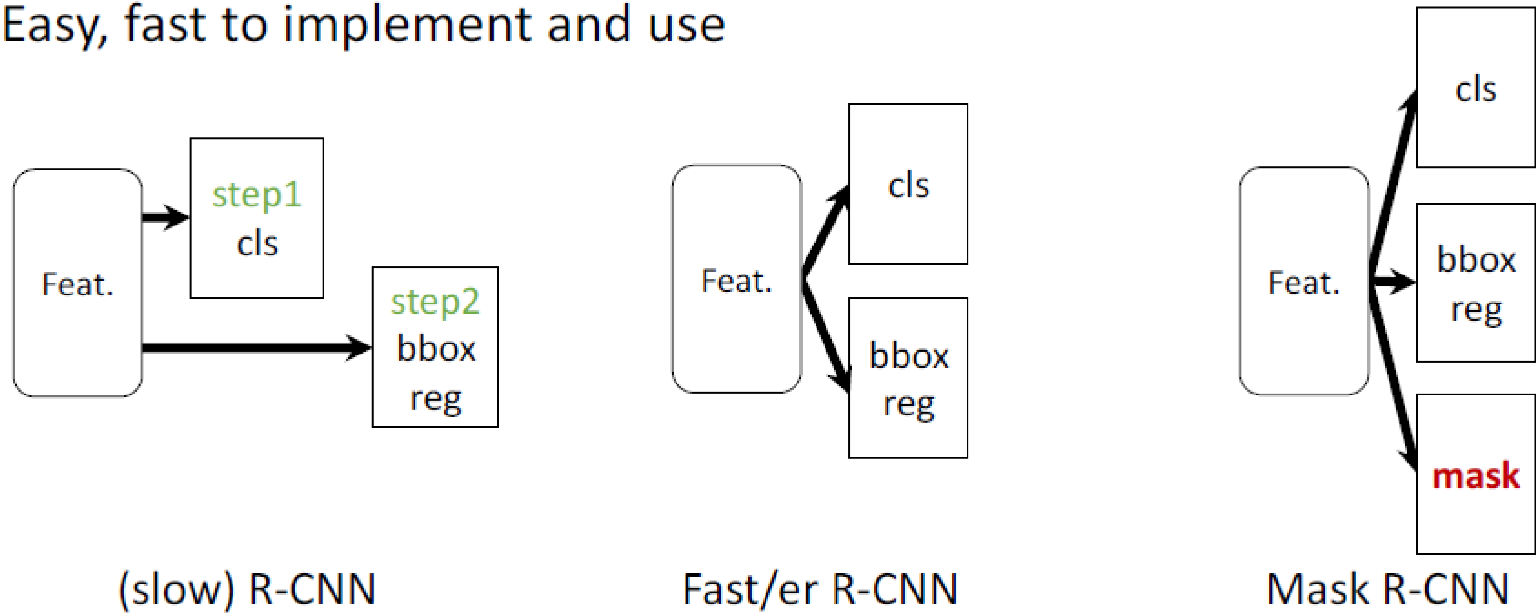
Slide by: Justin Johnson

Mask R-CNN

He et al, “Mask R-CNN”, ICCV 2017

What is Mask R-CNN: Parallel Heads

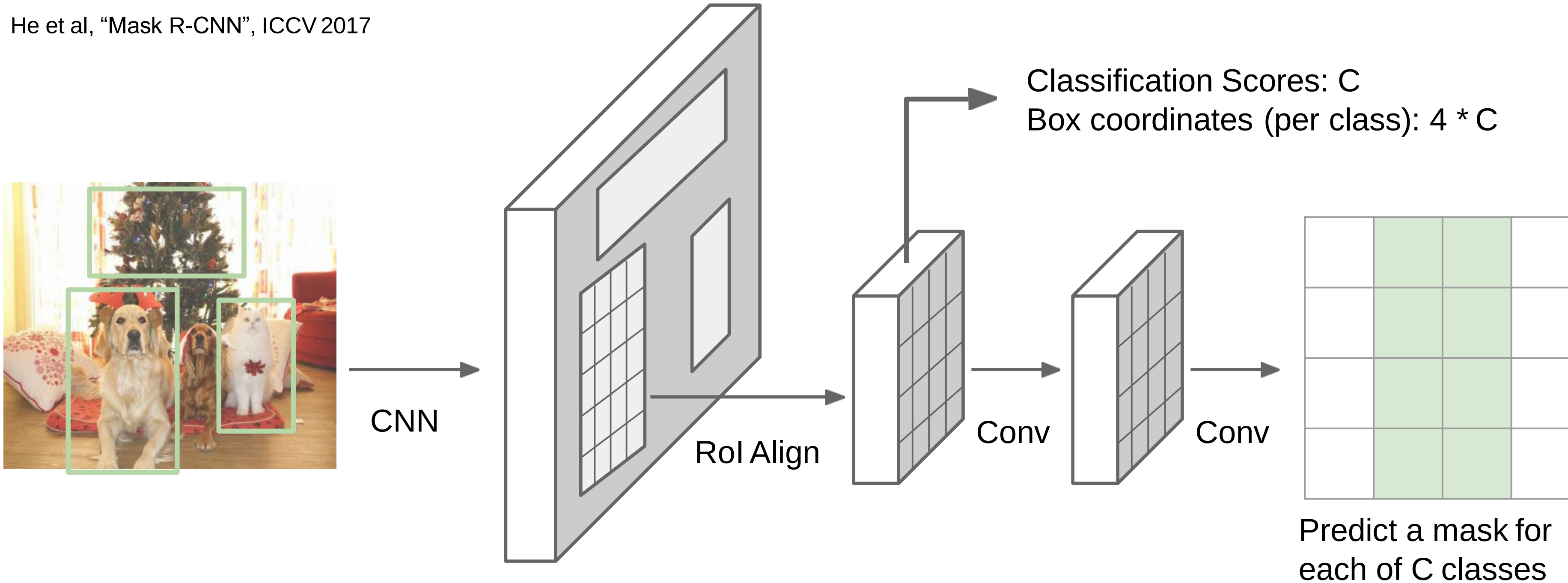
- Easy, fast to implement and use

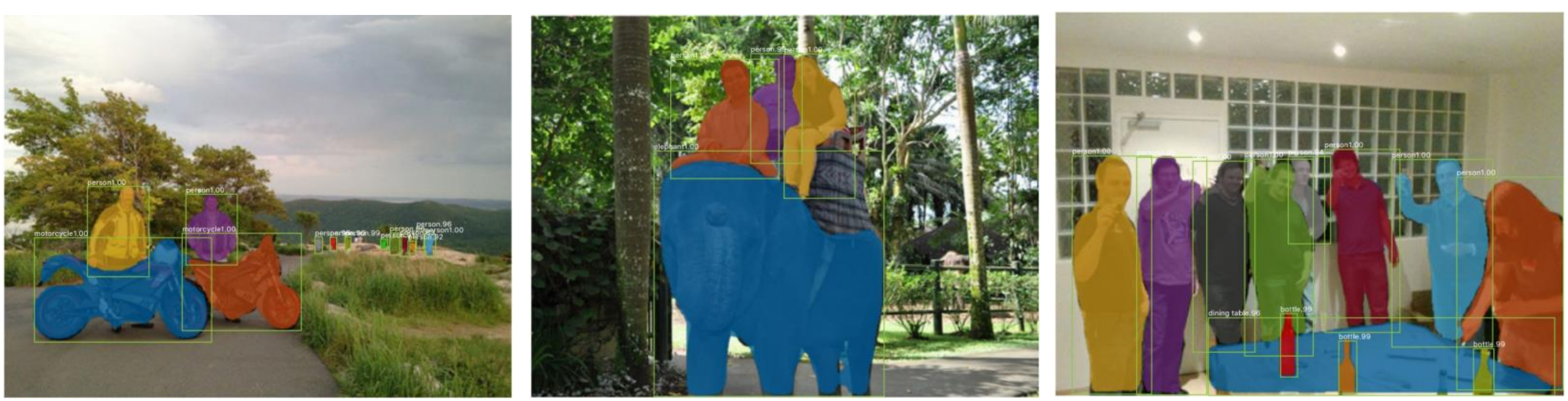


Slide by: Kaiming He

Mask R-CNN

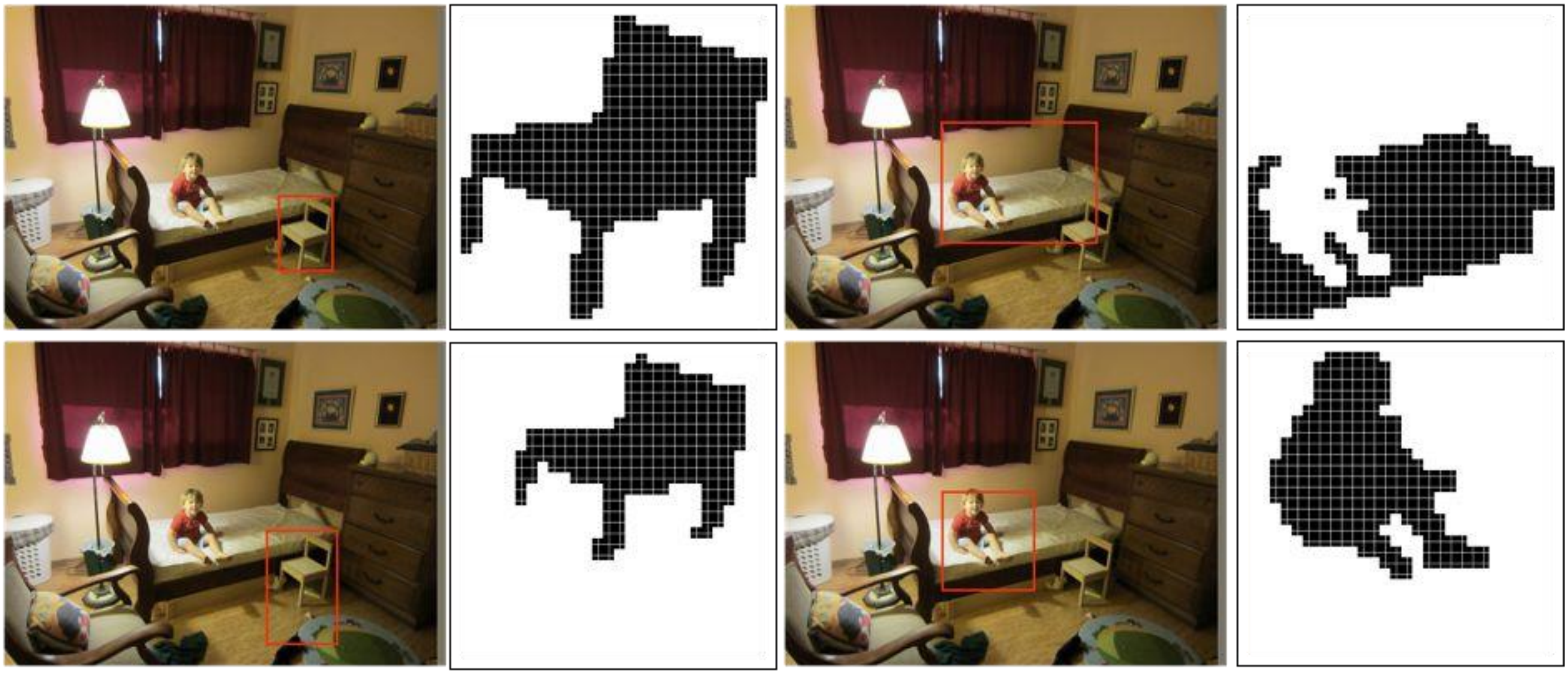
He et al, “Mask R-CNN”, ICCV 2017





Adapted from Justin Johnson

Mask R-CNN: Example Mask Training Targets



Slide by: Justin Johnson

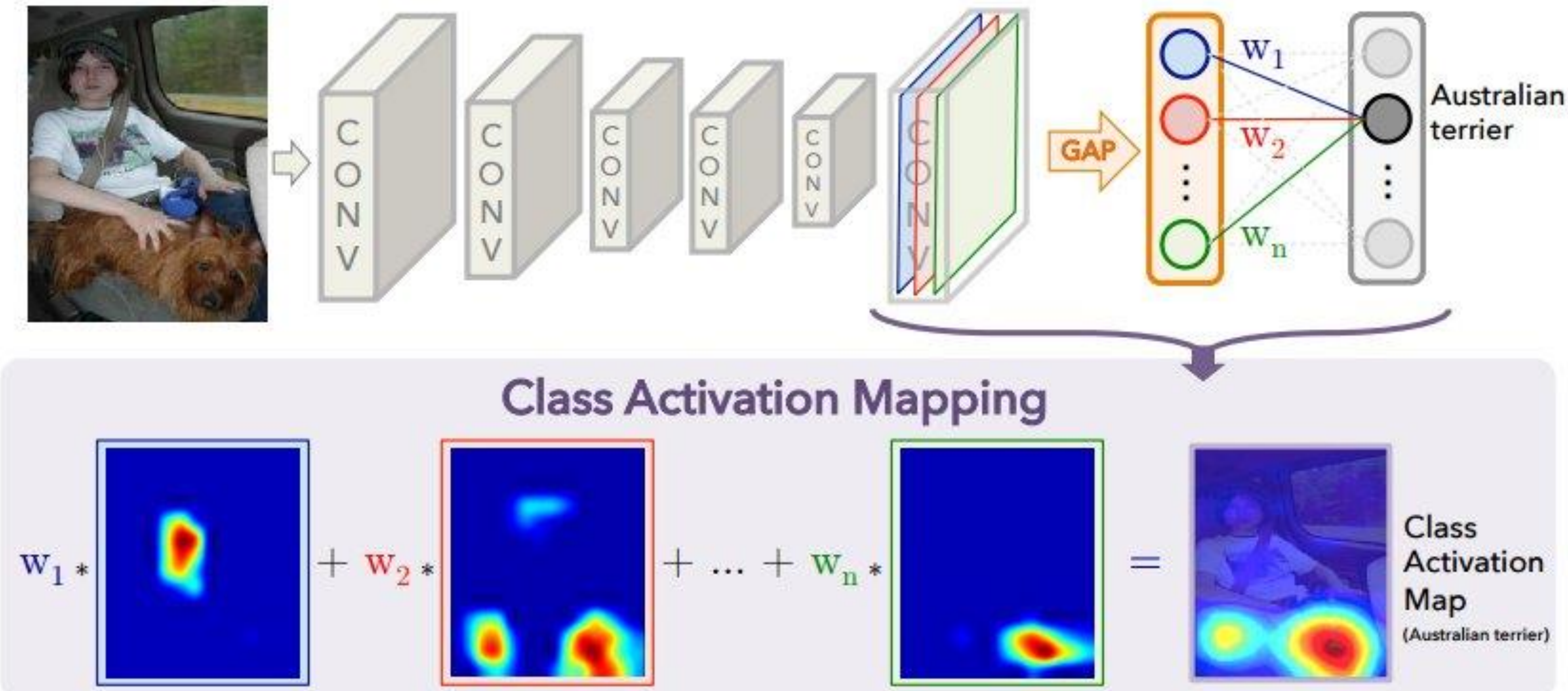
Plan for this lecture

- Fully supervised detection
  - Pre-CNN: Deformable part models
  - Detection with region proposals: R-CNN, Fast/er R-CNN
  - Detection without region proposals: YOLO
  - Semantic and instance segmentation: FCN, Mask R-CNN
- Weak or out-of-domain supervision
  - Weakly supervised object detection
  - Domain adaptation

What if no bounding boxes to train?

- Weakly supervised object detection
  - Image-level class labels
  - Image-level captions

Class activation maps



Zhou et al., “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class activation maps

- Let $f_k(x, y)$ be the activation in the k-th map at location (x, y)
- Global average pooling: $F^k = \Sigma_{x,y} f_k(x, y)$
- Input to softmax is $S_c = \Sigma_k w^c_k F^k$ where w^c_k is the weight for class c and map k

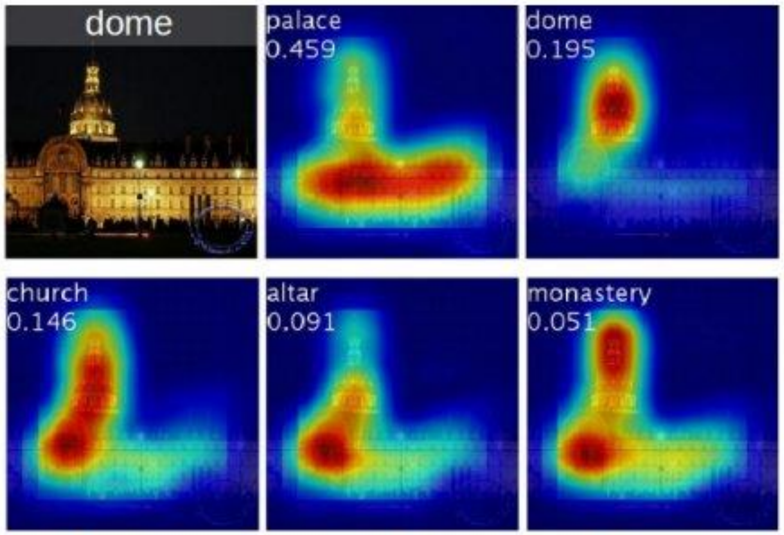
$$S_c = \sum_k w_k^c \sum_{x,y} f_k(x,y) = \sum_{x,y} \sum_k w_k^c f_k(x,y)$$

- Map for class c:

$$M_c(x,y) = \sum_k w_k^c f_k(x,y)$$

Zhou et al., “Learning Deep Features for Discriminative Localization”, CVPR 2016

Class activation maps



Class activation maps of top 5 predictions

Class activation maps for one object class

Table 3. Localization error on the ILSVRC test set for various weakly- and fully- supervised methods.

Method	supervision	top-5 test error
GoogLeNet-GAP (heuristics)	weakly	**37.1**
GoogLeNet-GAP	weakly	42.9
Backprop [23]	weakly	46.4
GoogLeNet [25]	full	26.7
OverFeat [22]	full	29.9
AlexNet [25]	full	34.2

Zhou et al., "Learning Deep Features for Discriminative Localization", CVPR 2016

Class activation maps

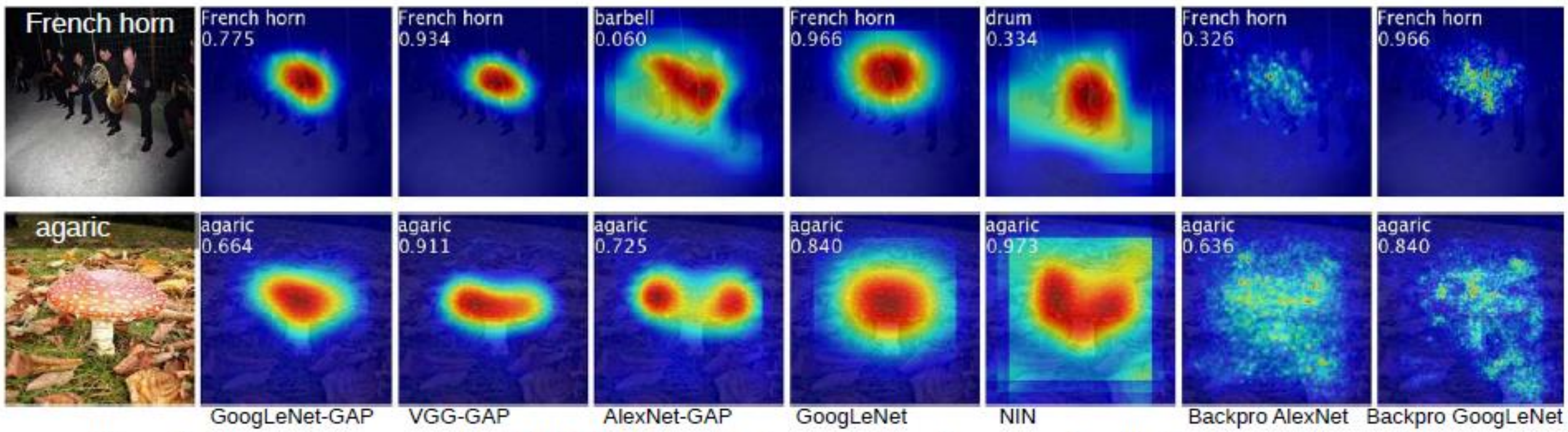


Figure 5. Class activation maps from CNN-GAPs and the class-specific saliency map from the backpropagation methods.

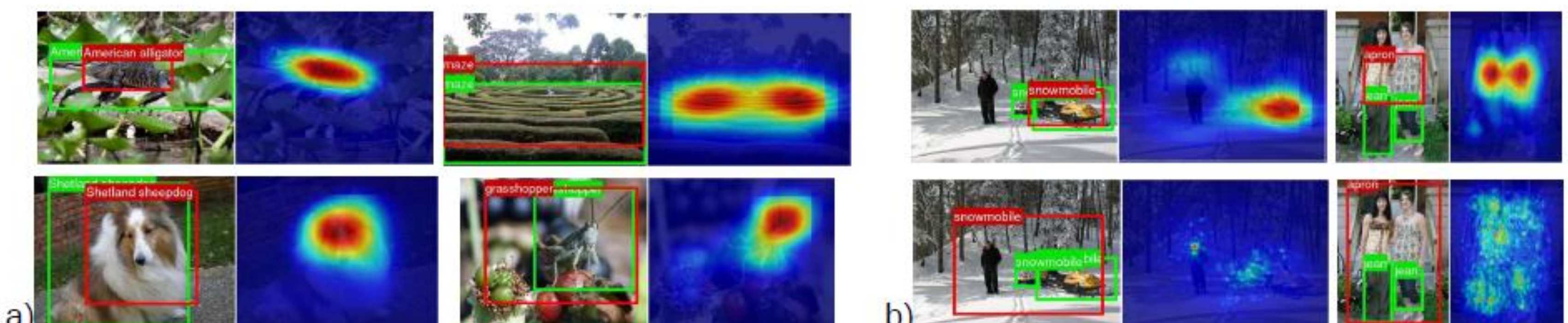
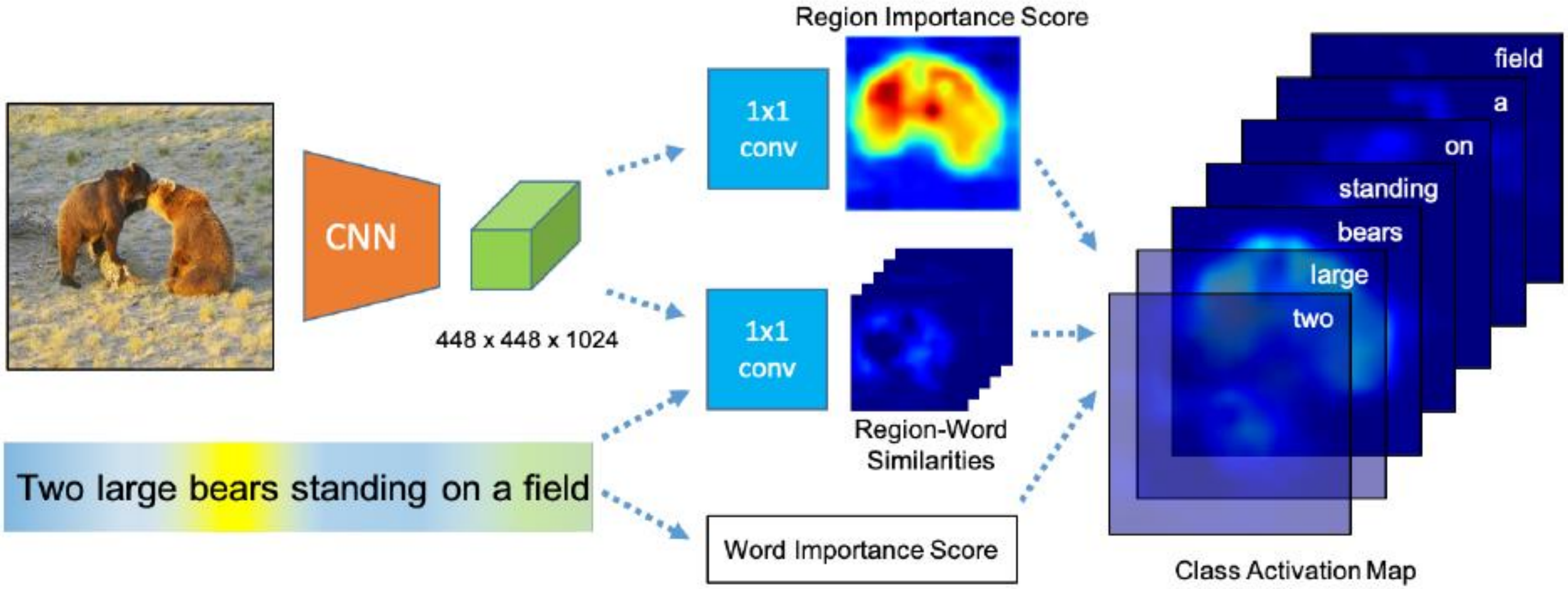


Figure 6. a) Examples of localization from GoogleNet-GAP. b) Comparison of the localization from GooleNet-GAP (upper two) and the backpropagation using AlexNet (lower two). The ground-truth boxes are in green and the predicted bounding boxes from the class activation map are in red.

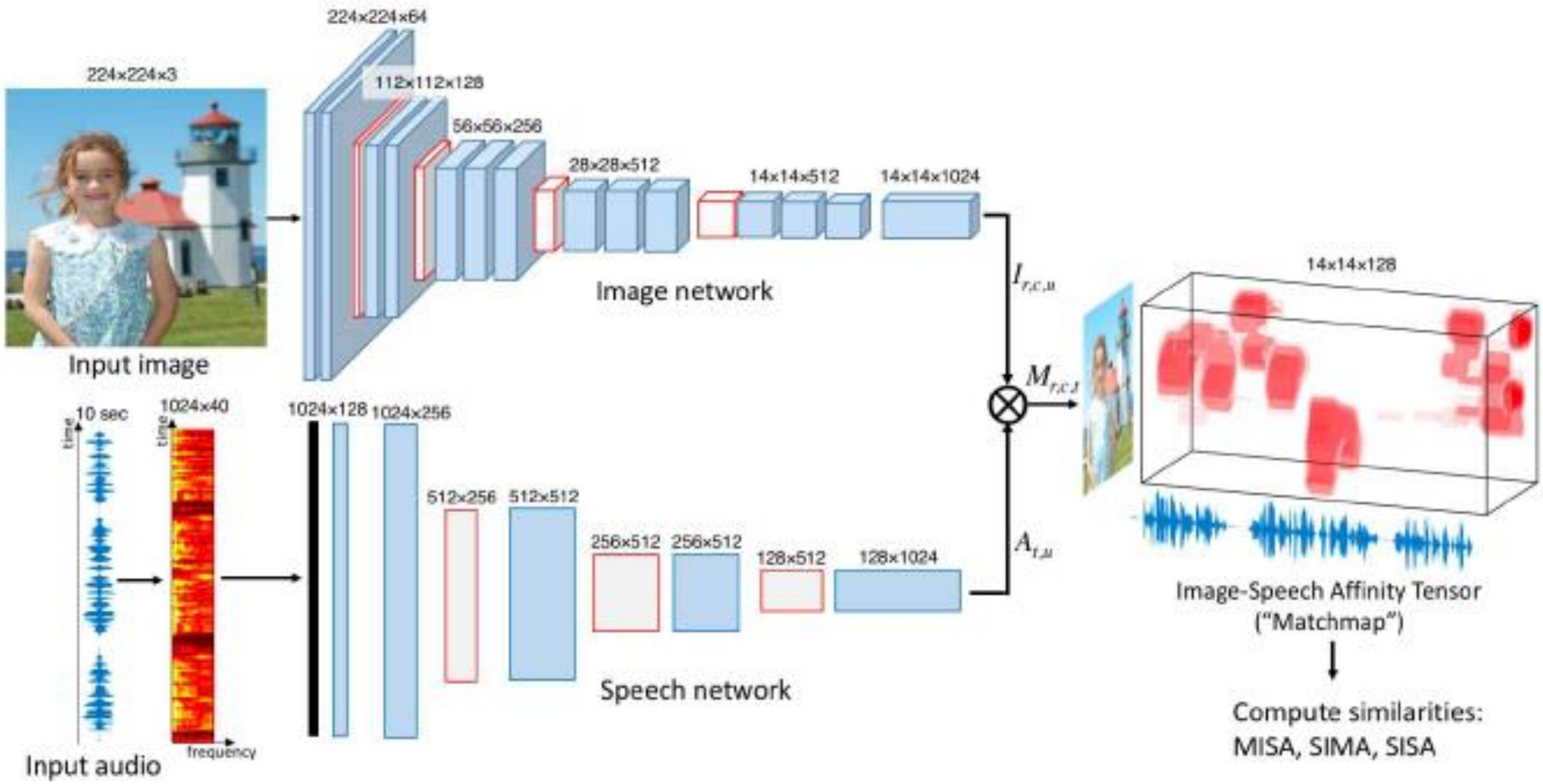
Zhou et al., “Learning Deep Features for Discriminative Localization”, CVPR 2016

Localization from captions



Ye et al., “Learning to discover and localize visual objects with open vocabulary”, arxiv 2018

Localization from sound



Harwath et al., "Jointly Discovering Visual Objects and Spoken Words from Raw Sensory Input", ECCV 2018

Localization from sound

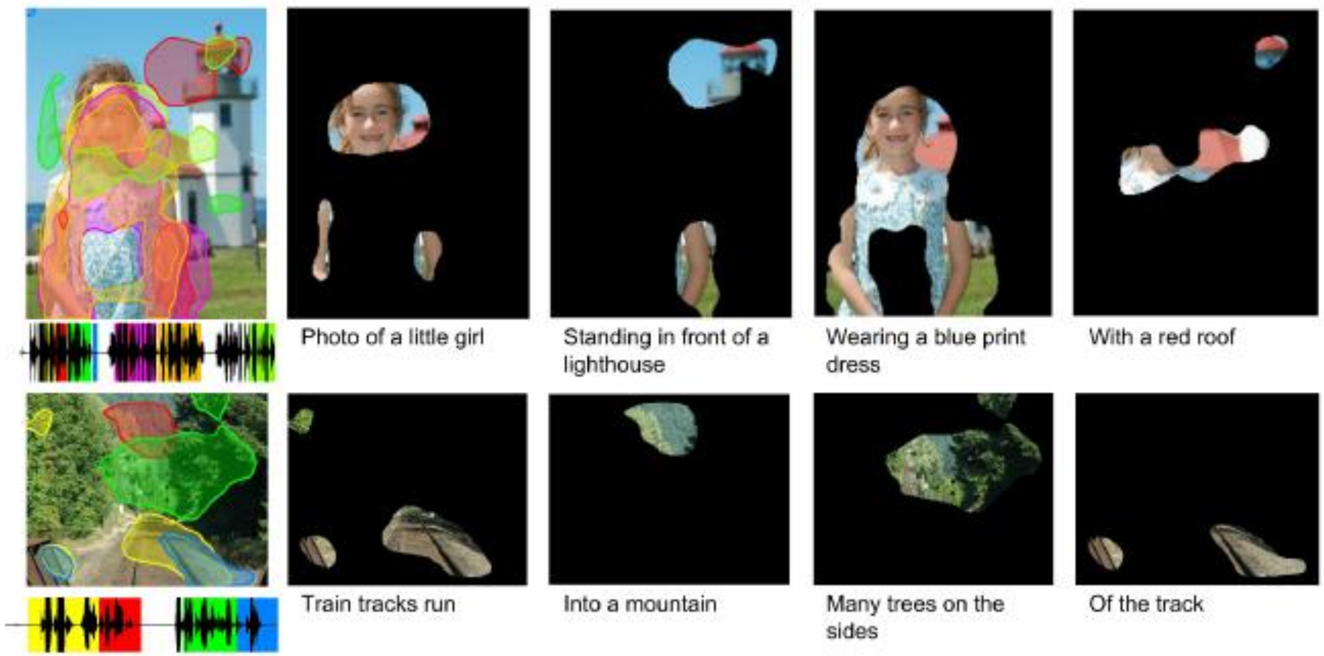
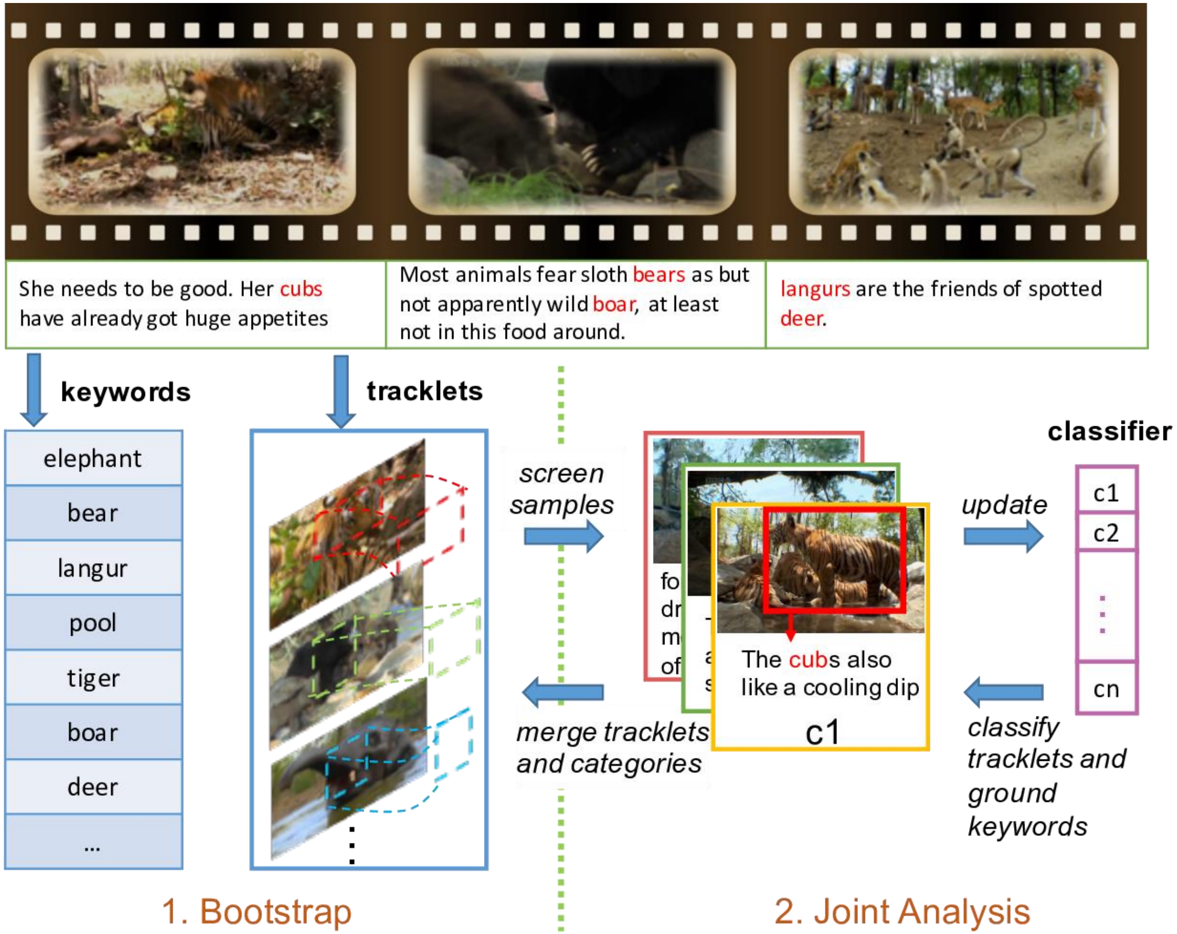


Fig. 7: On the left are shown two images and their speech signals. Each color corresponds to one connected component derived from two matchmaps from a fully random MISA network. The masks on the right display the segments that correspond to each speech segment. We show the caption words obtained from the ASR transcriptions below the masks. Note that those words were never used for learning, only for analysis.

Harwath et al., "Jointly Discovering Visual Objects and Spoken Words from Raw Sensory Input", ECCV 2018

Detection from documentaries

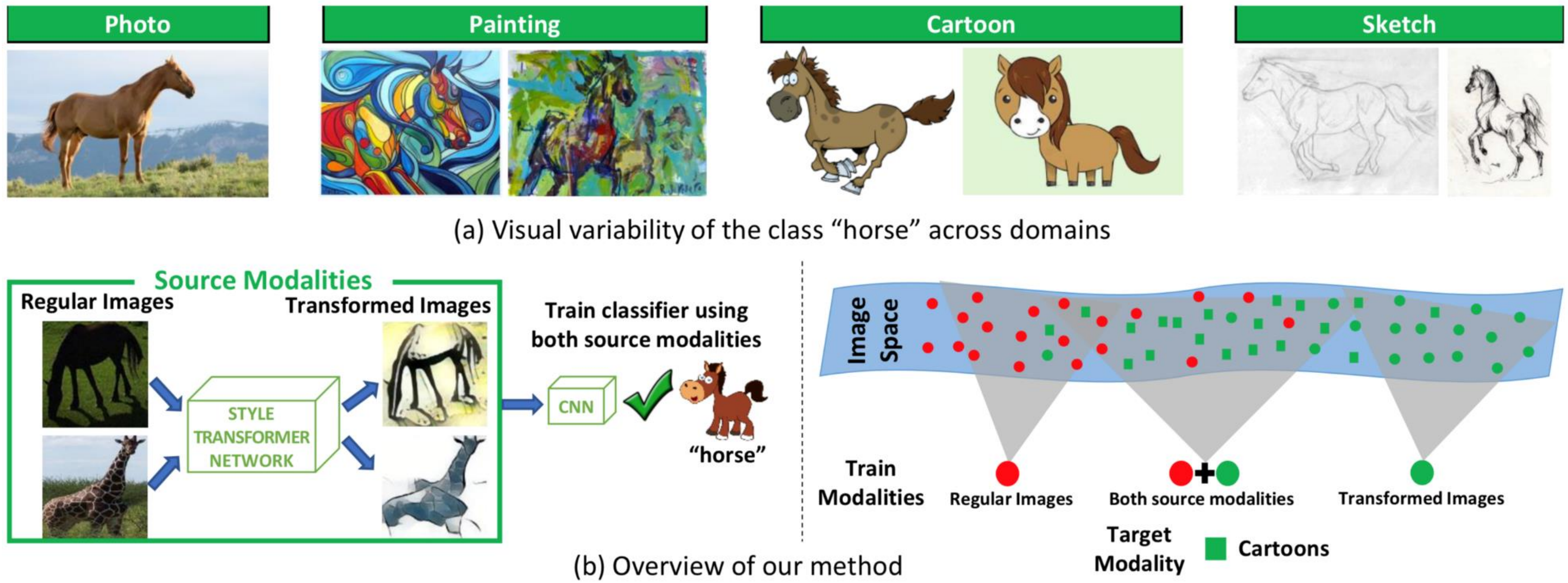


Chen et al., “Discover and Learn New Objects from Documentaries”, CVPR 2017

What if test data very diff from train?

- Adapt detection models

Adapting classifiers



Thomas and Kovashka, "Artistic Object Recognition by Unsupervised Style Adaptation", ACCV 2018

Adapting classifiers

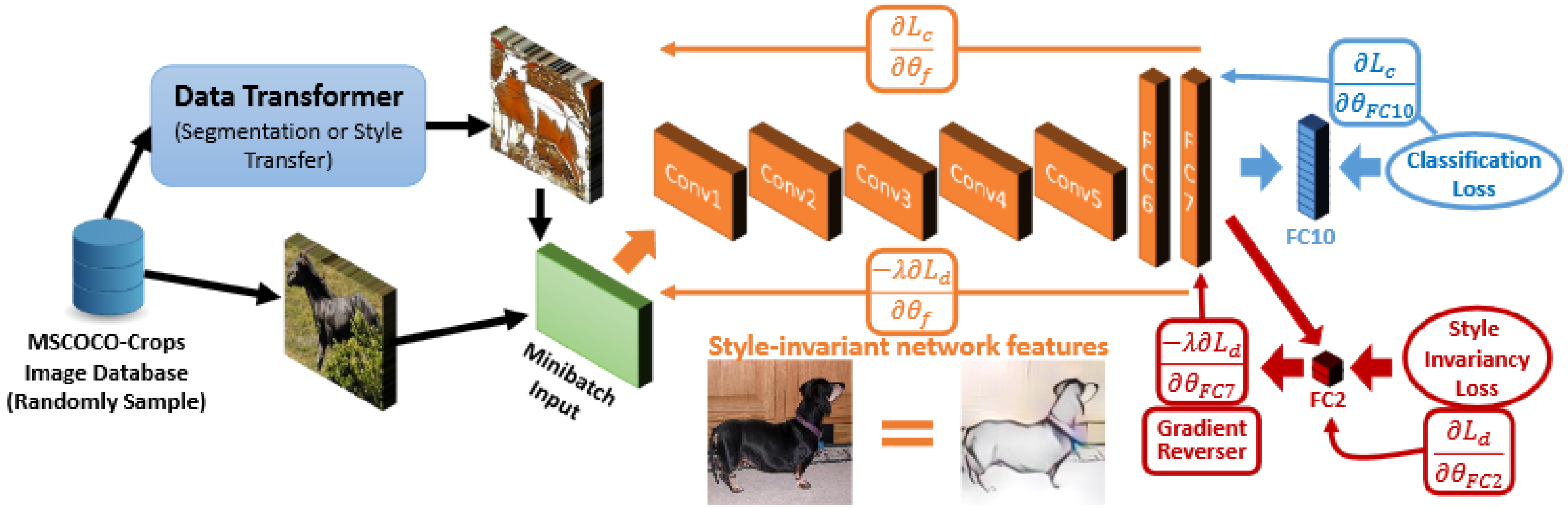


Fig. 2. Training with multiple modalities and style-invariance constraint. We train networks on real and synthetic data. We show an example of style transfer transforming photos into labeled synthetic cartoons. The style-invariance loss trains the FC2 layer to predict which modality the image came from. During backpropagation, we reverse its gradient before propagating it to the layers used by both classifiers. This encourages those layers to learn style-invariant features.

Thomas and Kovashka, “Artistic Object Recognition by Unsupervised Style Adaptation”, ACCV 2018

Adapting detectors

Figure 1. **Illustration of different datasets for autonomous driving:** From top to bottom-right, example images are taken from: *KITTI*[17], *Cityscapes*[5], *Foggy Cityscapes*[49], *SIM10K*[30]. Though all datasets cover urban scenes, images in those dataset vary in style, resolution, illumination, object size, *etc*. The visual difference between those datasets presents a challenge for applying an object detection model learned from one domain to another domain.

Chen et al., “Domain Adaptive Faster R-CNN for Object Detection in the Wild”, CVPR 2018

Adapting detectors

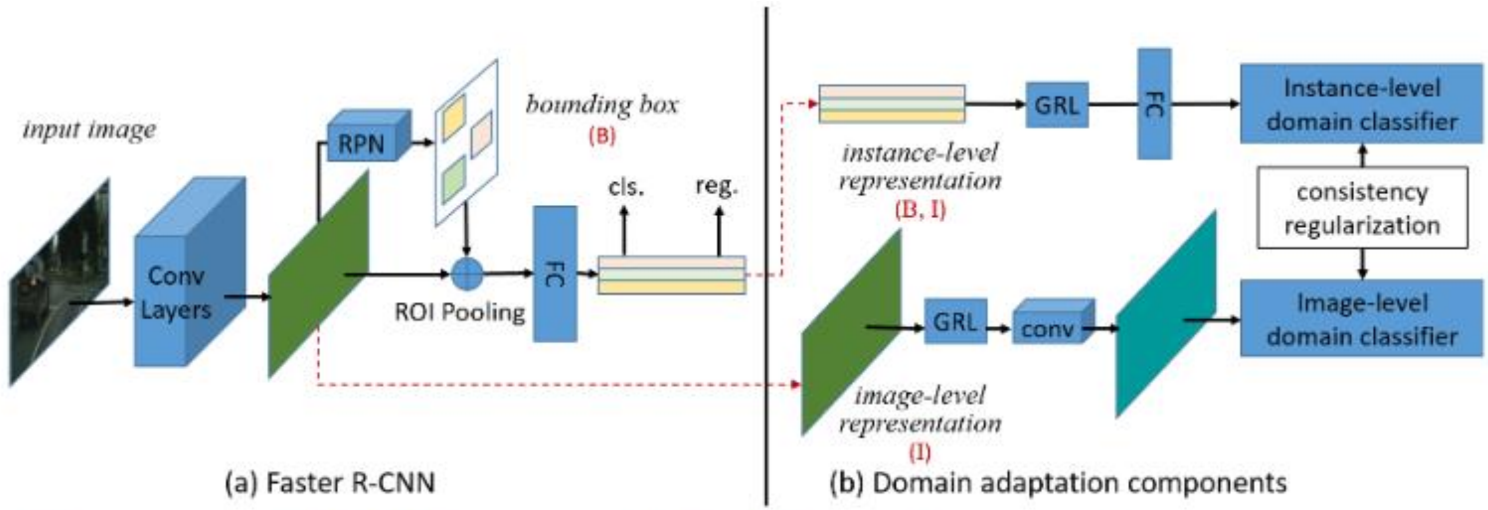


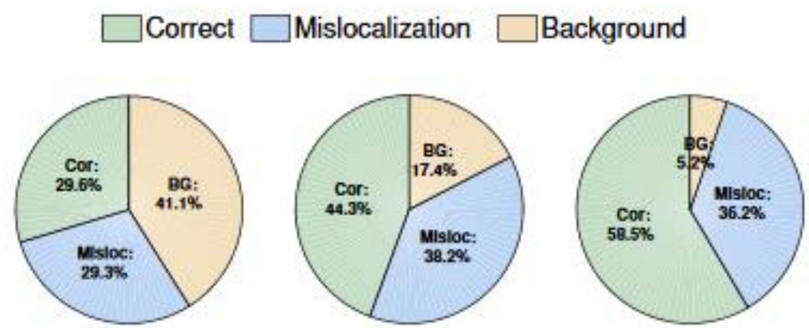
Figure 2. **An overview of our Domain Adaptive Faster R-CNN model:** we tackle the domain shift on two levels, the image level and the instance level. A domain classifier is built on each level, trained in an adversarial training manner. A consistency regularizer is incorporated within these two classifiers to learn a domain-invariant RPN for the Faster R-CNN model.

Chen et al., “Domain Adaptive Faster R-CNN for Object Detection in the Wild”, CVPR 2018

Adapting detectors

	img	ins	cons	car AP
Faster R-CNN				30.12
Ours	✓			33.03
		✓		35.79
	✓	✓		37.86
	✓	✓	✓	**38.97**

Table 1. The average precision (AP) of *Car* on the *Cityscapes* validation set. The models are trained using the *SIM 10k* dataset as the source domain and the *Cityscapes* training set as the target domain. *img* is short for *image-level alignment*, *ins* for *instance-level alignment* and *cons* is short for our *consistency loss*



(a) Faster RCNN (b) Ours (Ins. Only) (c) Ours (Img Only)

Figure 3. Error Analysis of Top Ranked Detections

Chen et al., "Domain Adaptive Faster R-CNN for Object Detection in the Wild", CVPR 2018