

# lecture 9: object detection

## deep learning for vision

Yannis Avrithis

Inria Rennes-Bretagne Atlantique

Rennes, Nov. 2018 – Jan. 2019



# outline

**background**

**two-stage detection**

**object parts and deformation**

**scale and feature pyramids\***

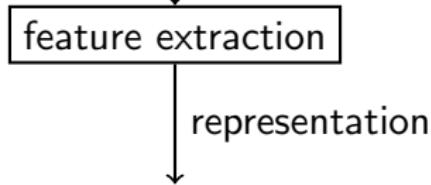
**one-stage detection**

# background

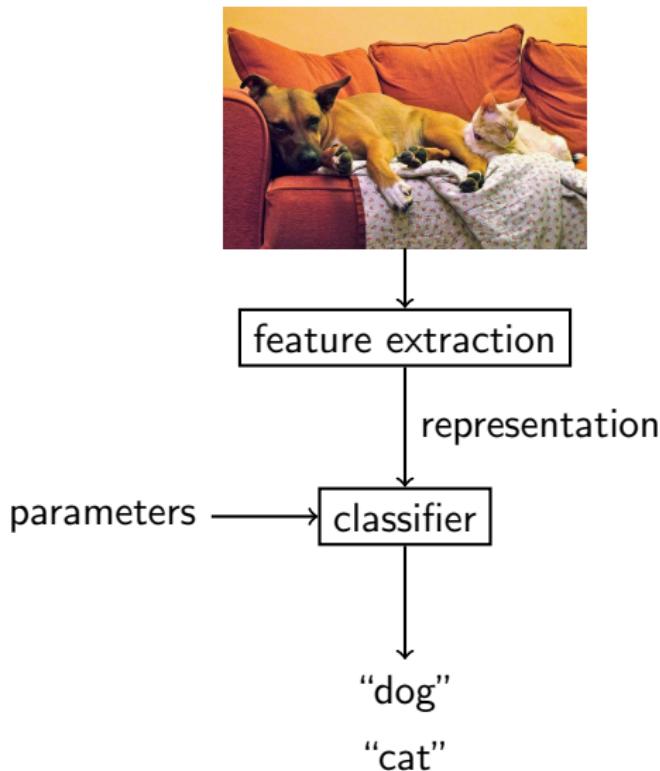
# data-driven approach



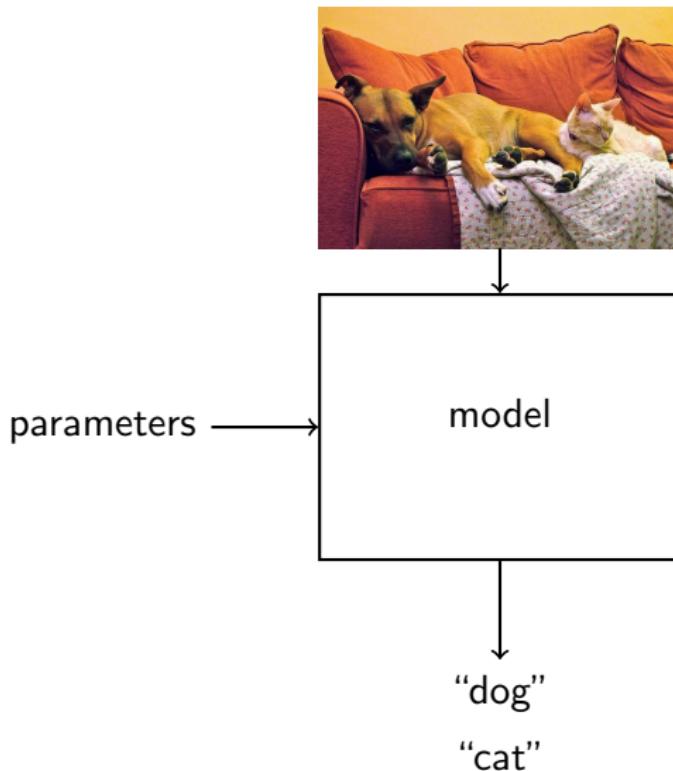
# data-driven approach



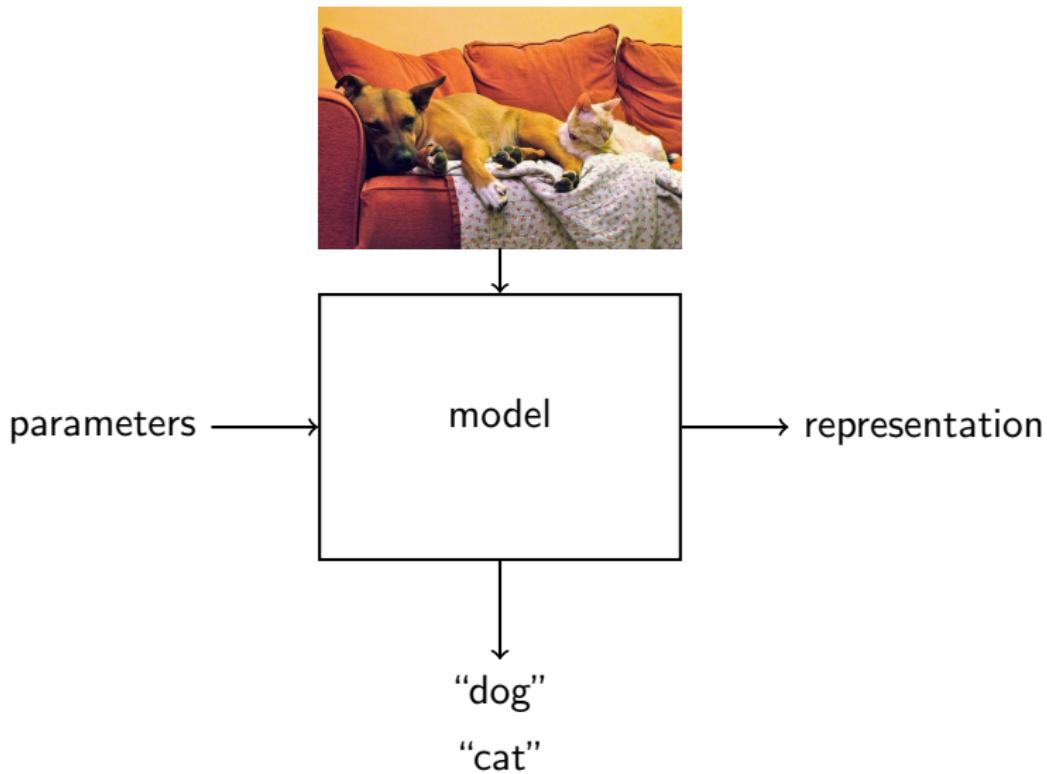
# data-driven approach



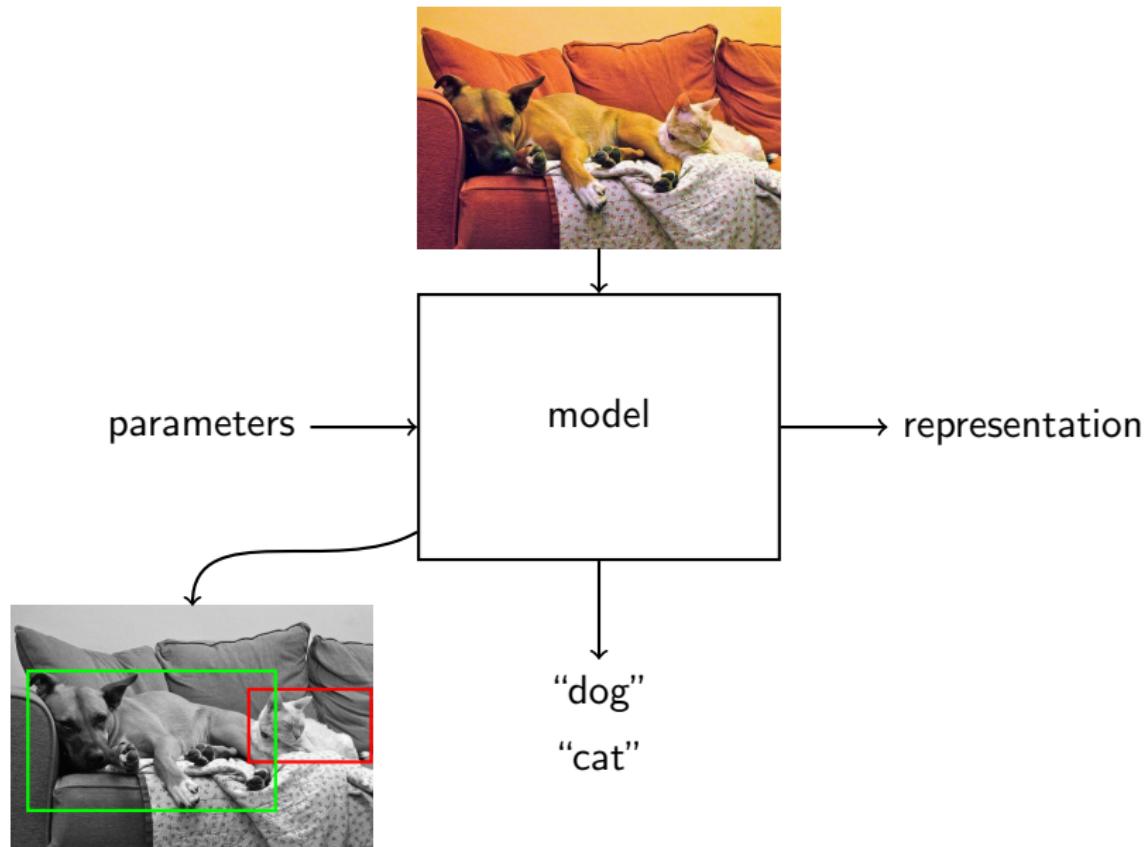
# data-driven approach



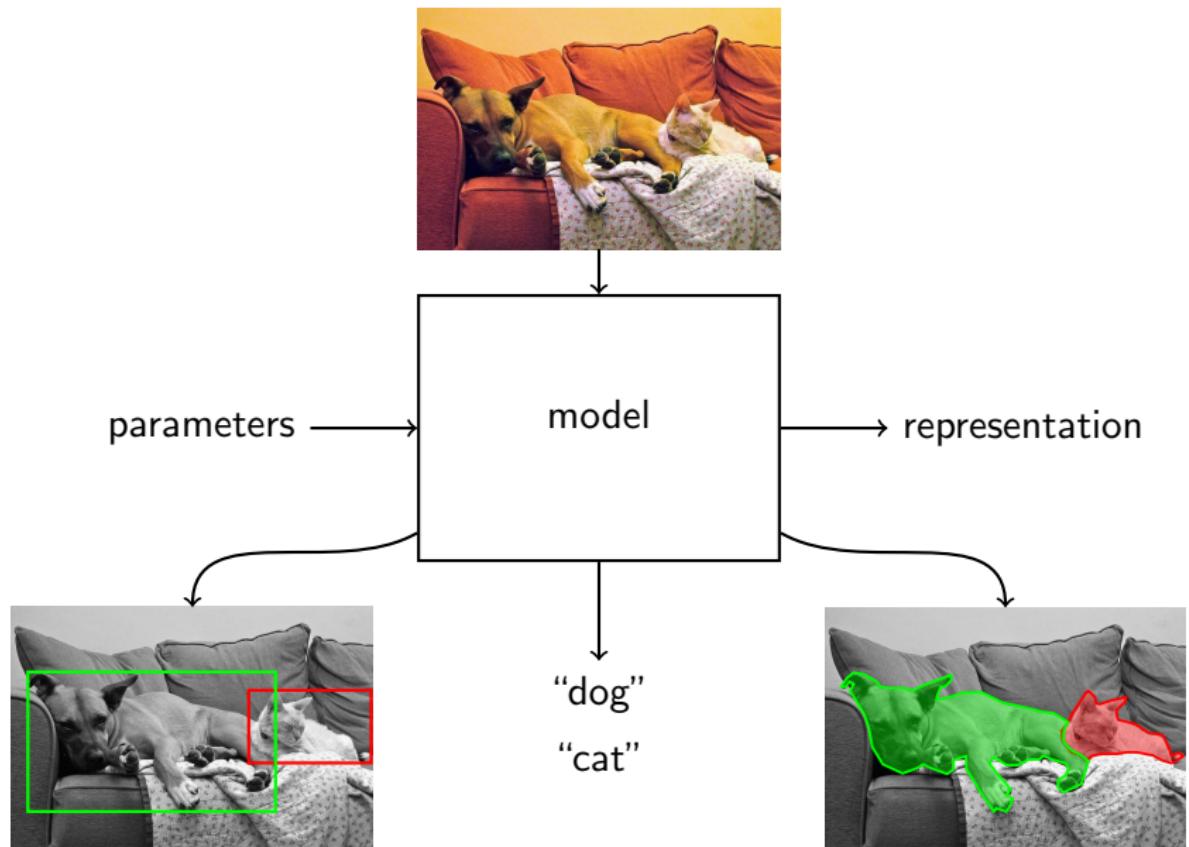
# data-driven approach



# data-driven approach



# data-driven approach



# beyond classification



object localization

classify + regress

bounding box  $(x, y, w, h)$

# beyond classification



**object localization**  
classify + regress  
bounding box  $(x, y, w, h)$



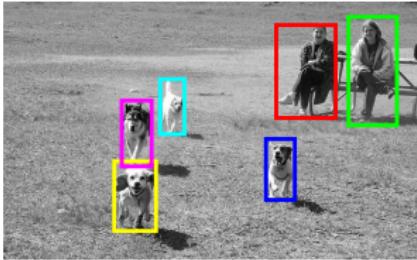
**semantic segmentation**  
pixel-wise classify

# beyond classification



## object localization

classify + regress  
bounding box  $(x, y, w, h)$



## object detection

per region: classify + regress  
bounding box  $(x, y, w, h)$

# beyond classification



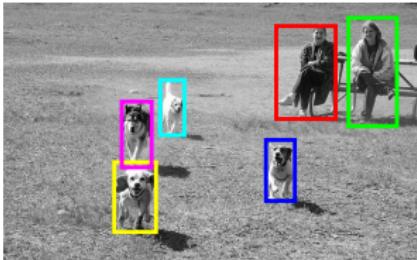
**object localization**

classify + regress  
bounding box  $(x, y, w, h)$



**semantic segmentation**

pixel-wise classify



**object detection**

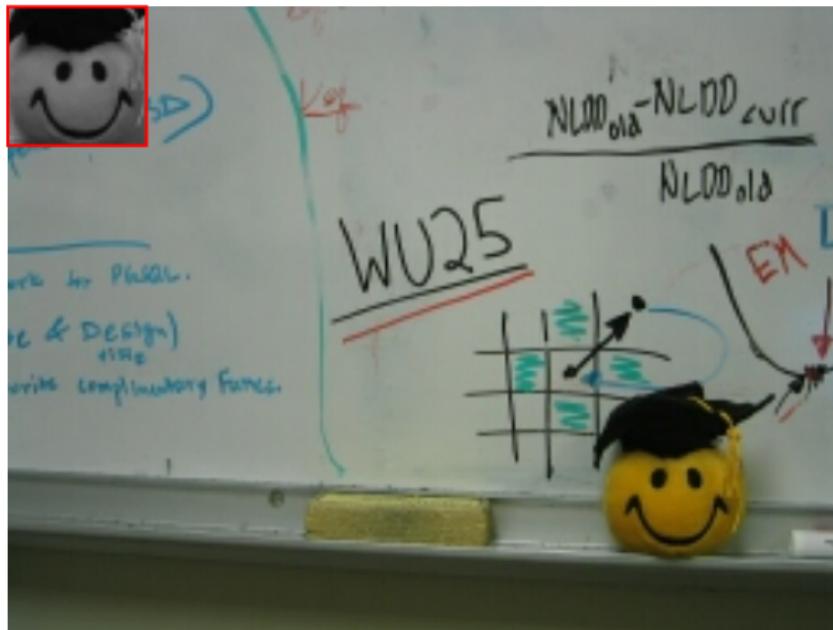
per region: classify + regress  
bounding box  $(x, y, w, h)$



**instance segmentation**

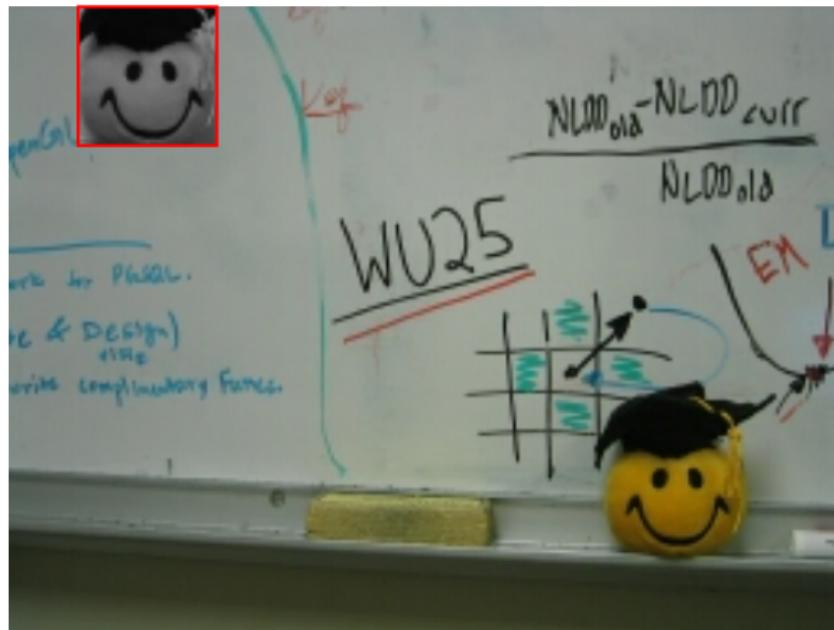
per region: pixel-wise classify

## template matching, or sliding window



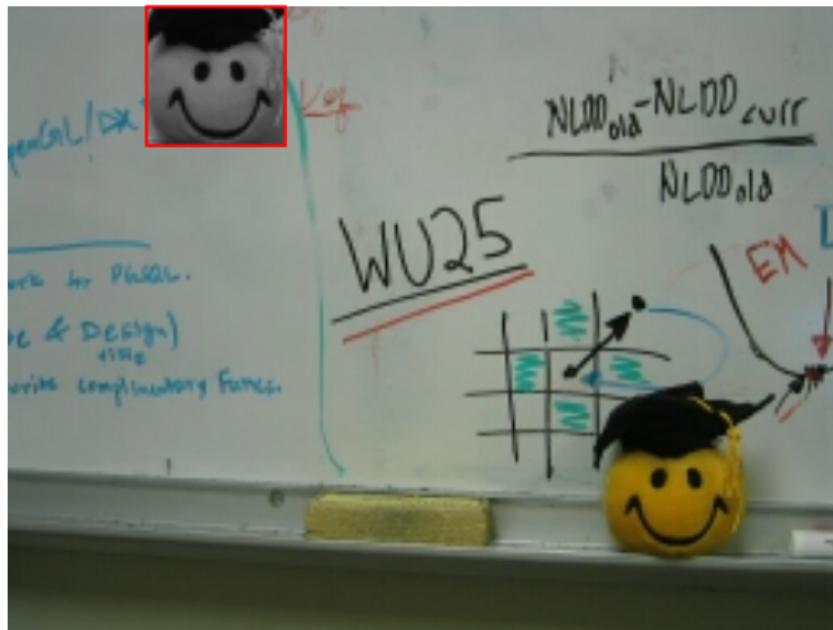
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



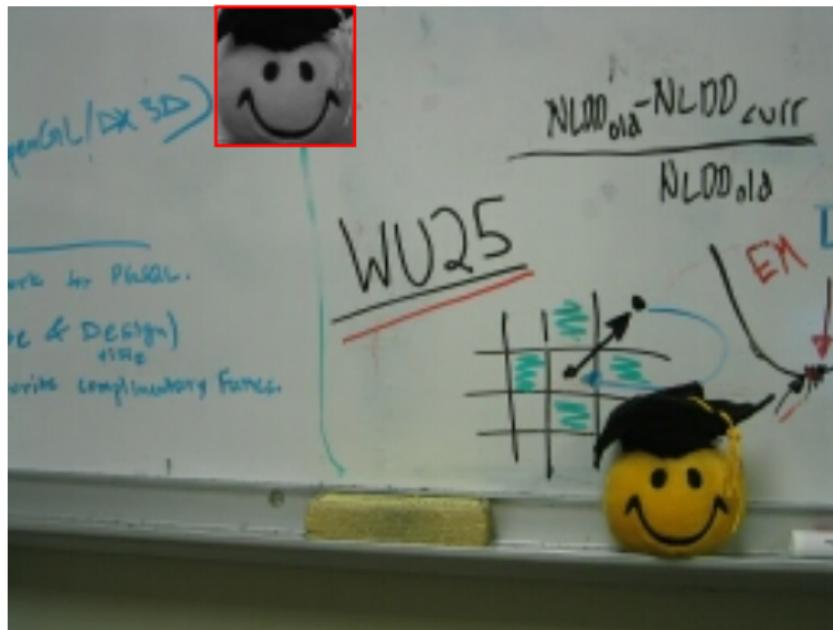
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



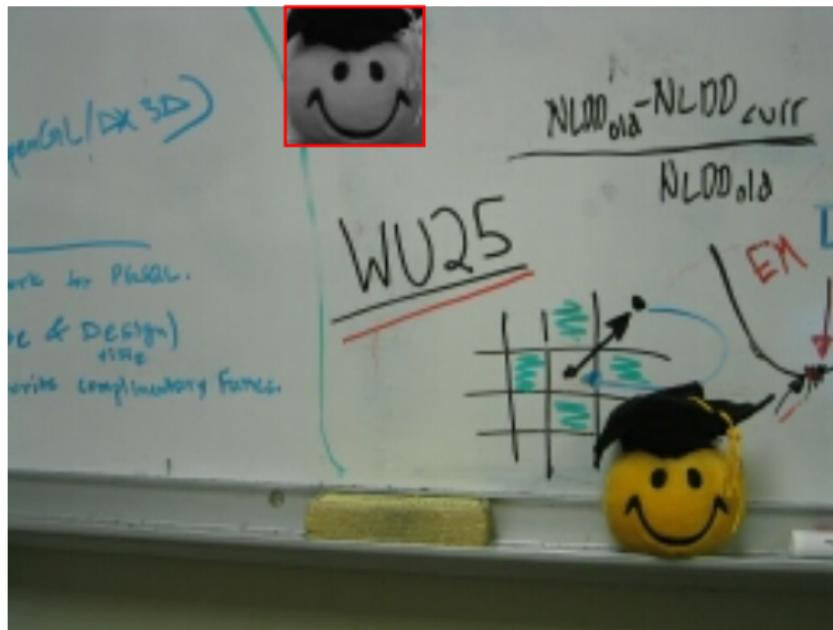
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



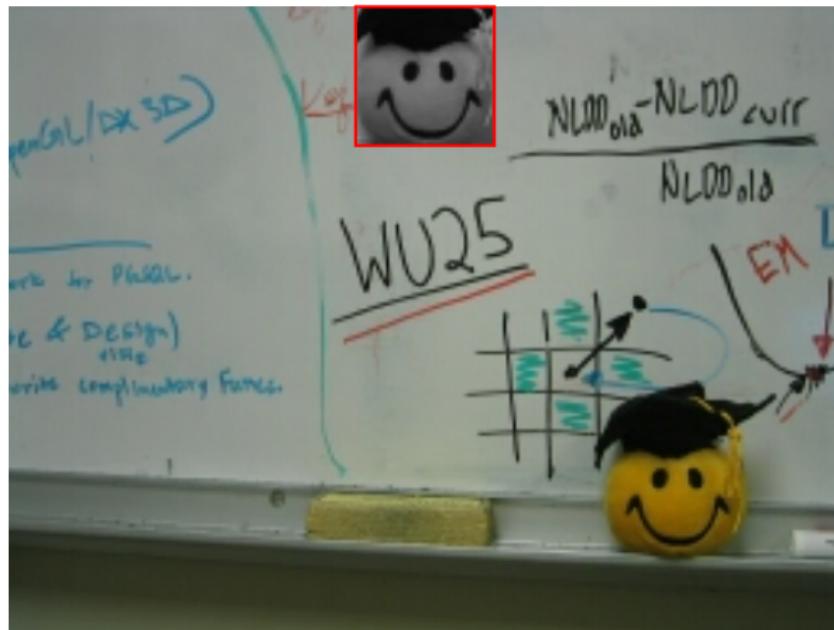
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



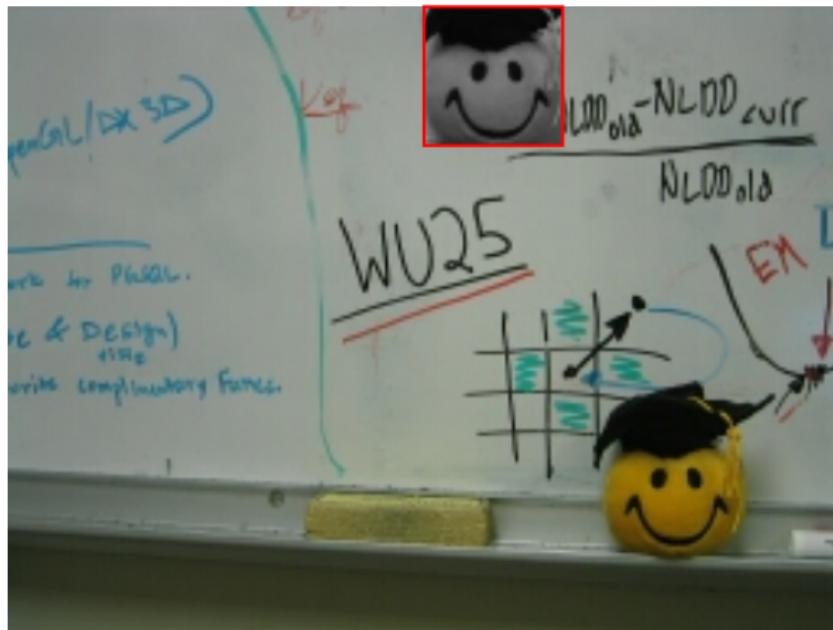
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



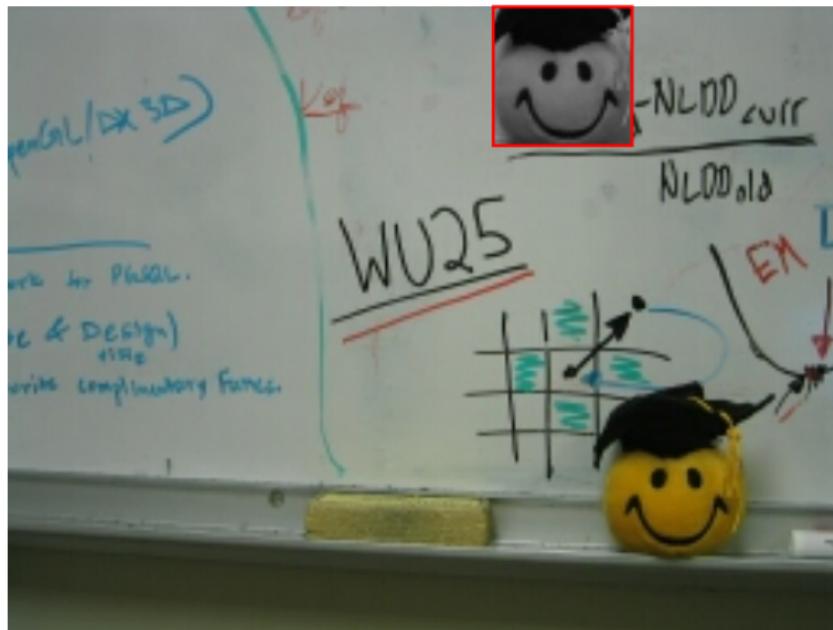
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



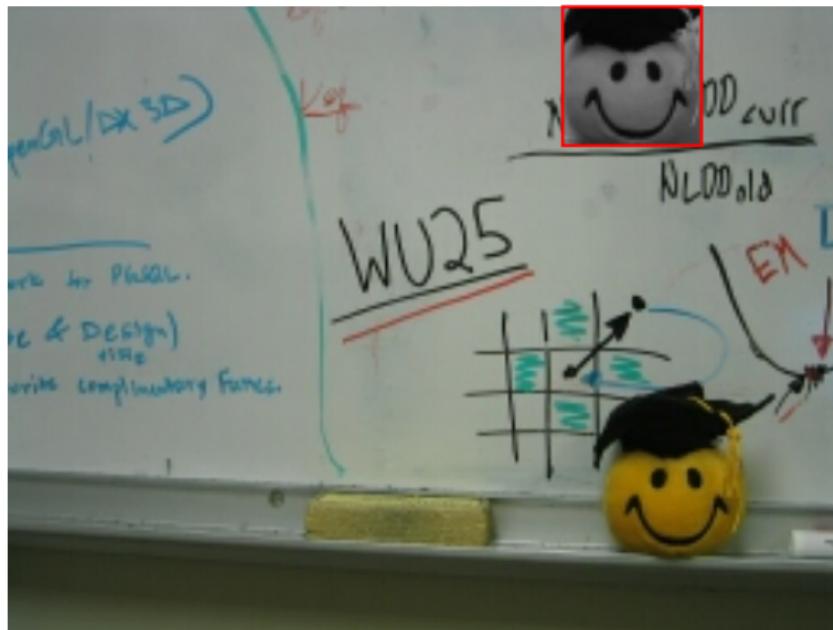
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



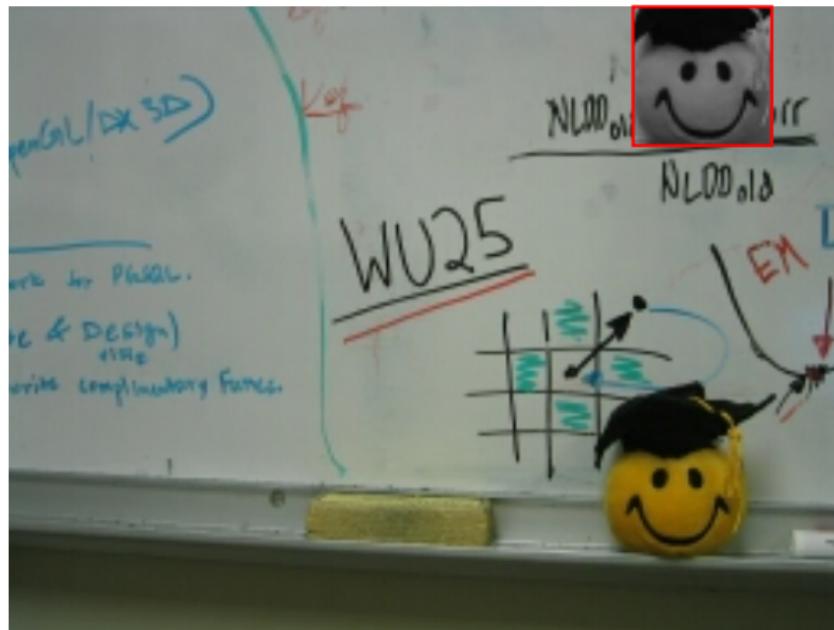
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



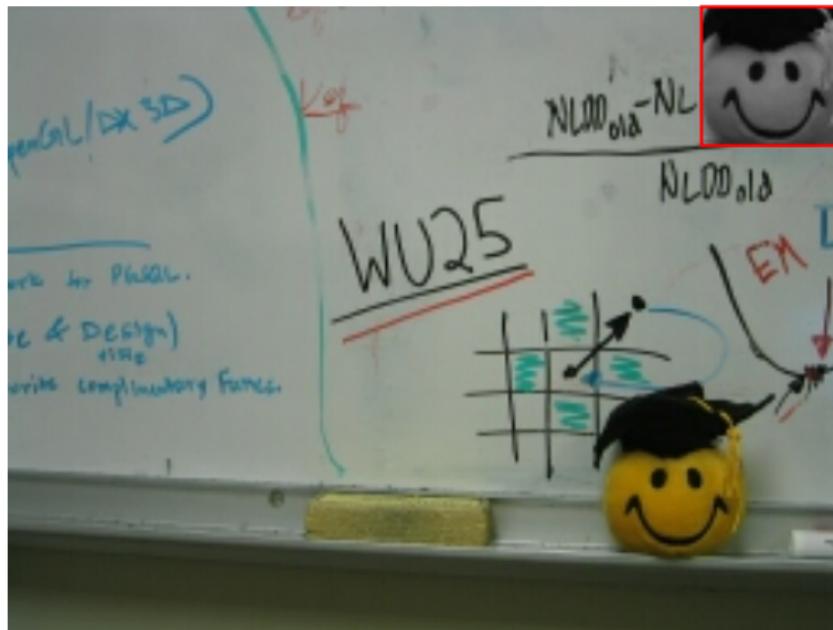
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



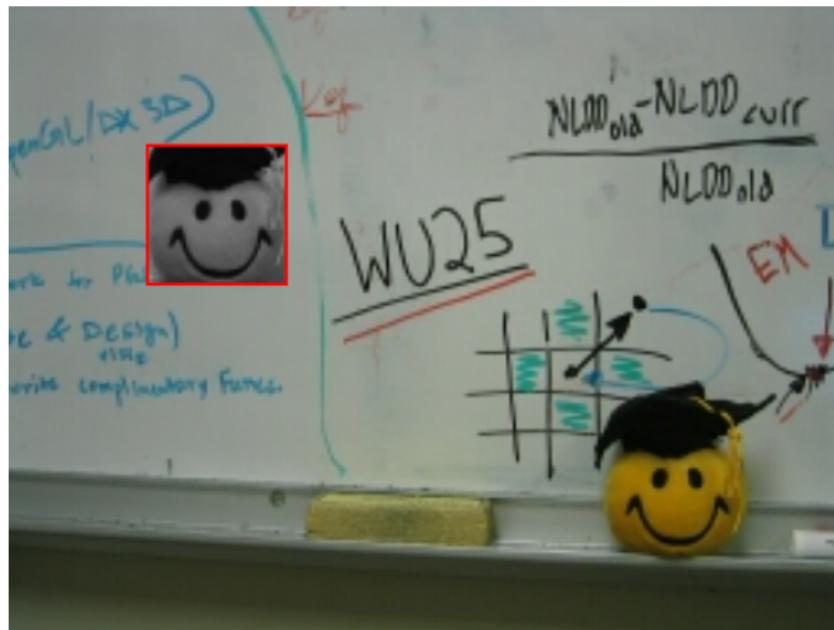
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



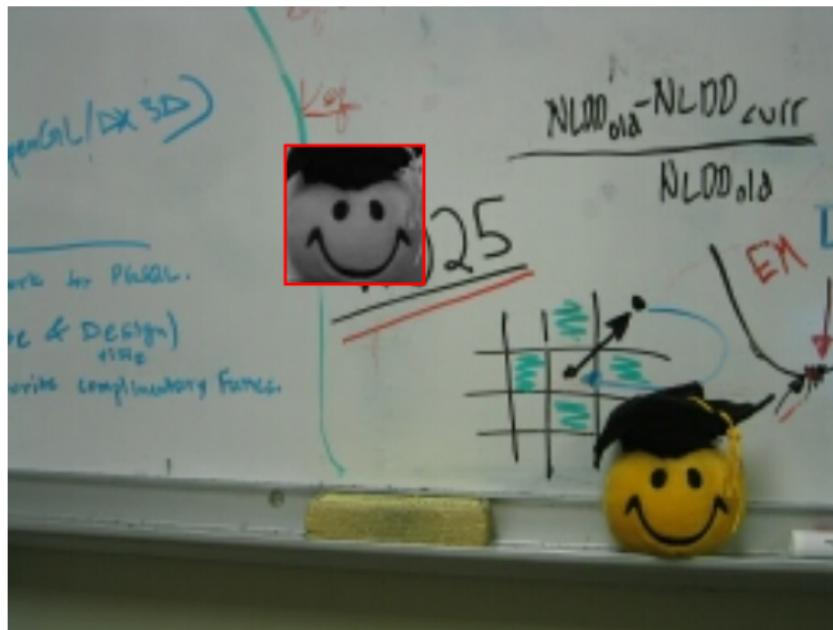
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



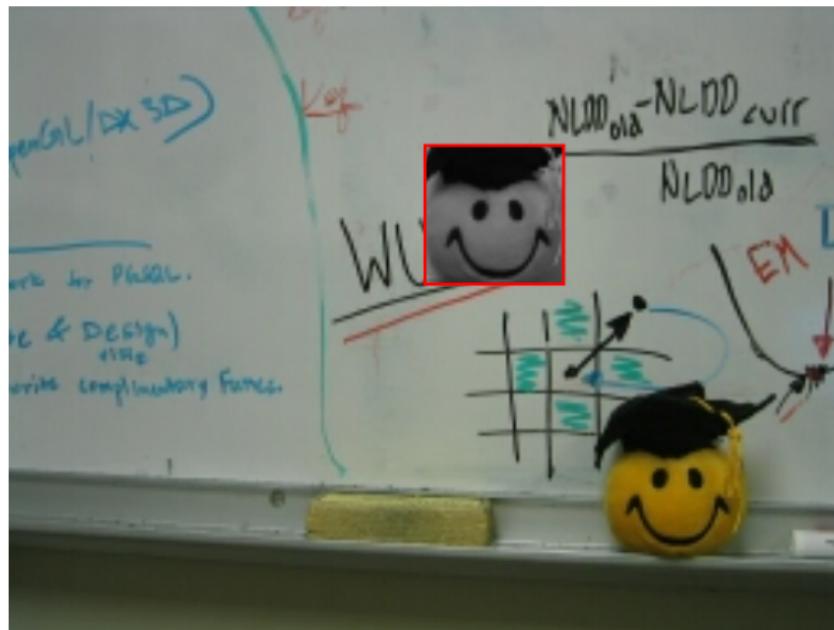
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



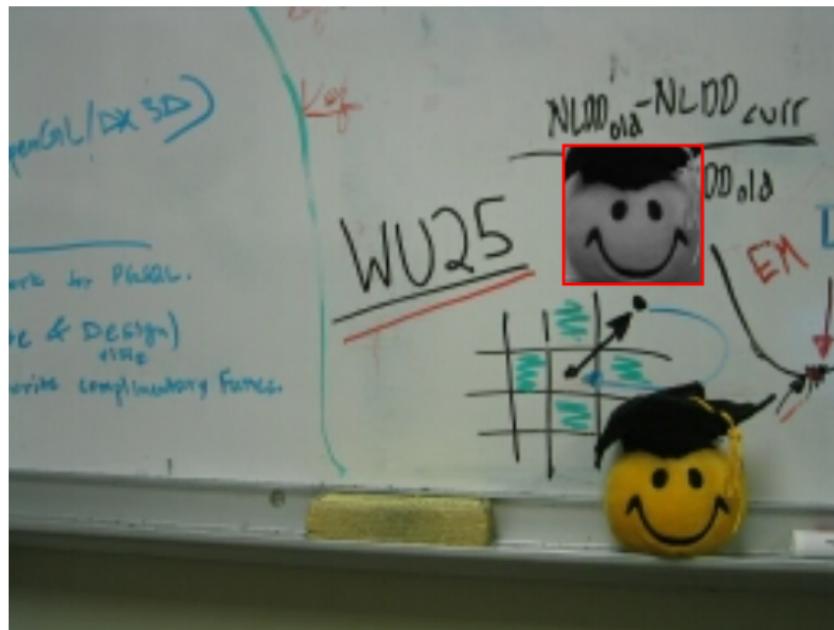
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



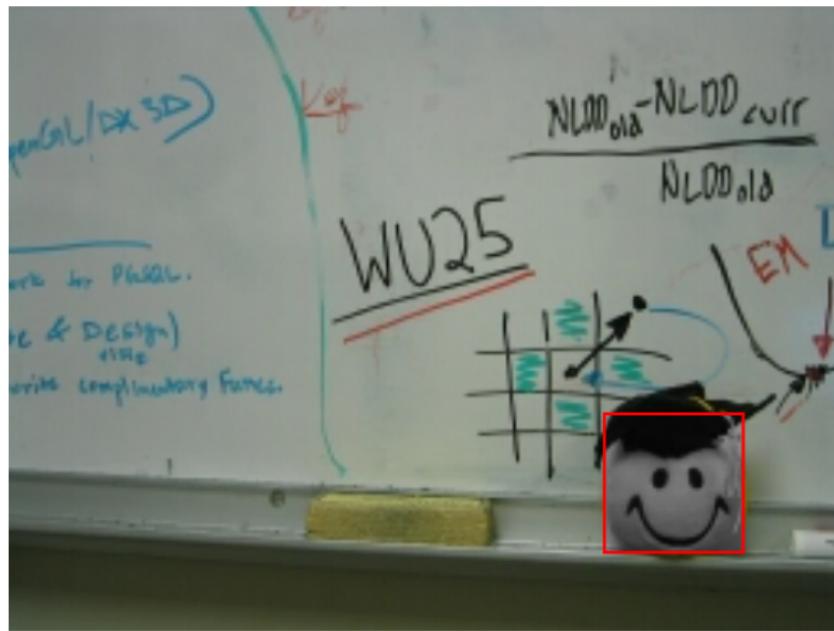
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score

## template matching, or sliding window



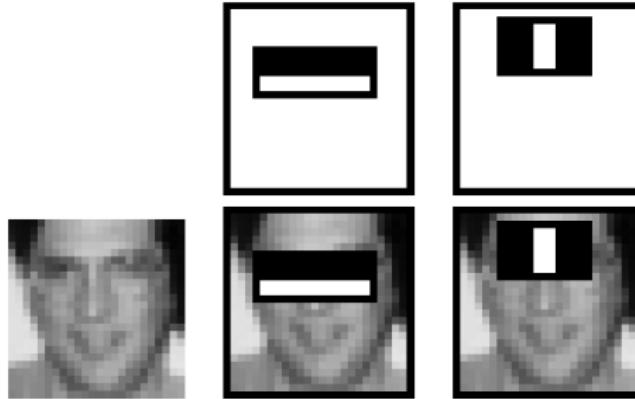
- slide template over image at multiple positions
- positions can be overlapping, or even **dense** (every pixel)
- seek maximum similarity score (e.g. cross-correlation)

## two problems

- to detect a given instance (template), a similarity score may be enough; but to detect an object of a given class, we need strong **features** and a good **classifier**
- with unknown position, scale and aspect ratio, the search space is 4-dimensional: to search **efficiently**, we need something better than exhaustive search

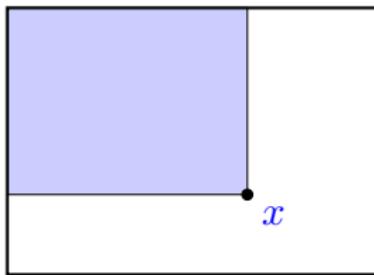
# real-time face detection

[Viola and Jones 2001]



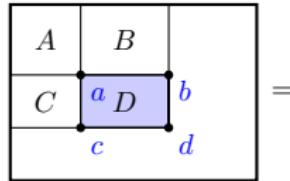
- millions of simple features exhaustively evaluated on integral image
- learning weak classifiers by AdaBoost
- classifier cascade provides a focus-of-attention mechanism

## integral image: construction



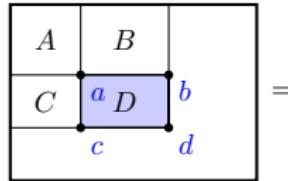
- given an image, precompute its sum over the rectangle with vertices the top-left corner and any point  $x$  in the image
- the collection of all sums is the **integral image**: it can be computed by one pass over the original image and takes the same space as the original image

## integral image: use

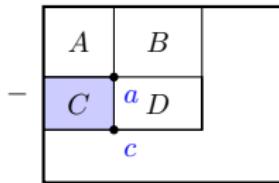
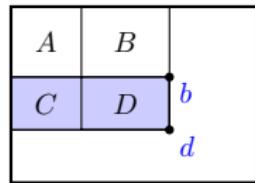


- then, the sum over any rectangle ( $D$ ) can be evaluated by 3 scalar operations on its vertices ( $a, b, c, d$ )

## integral image: use



=

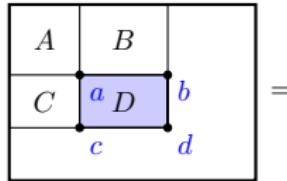


-

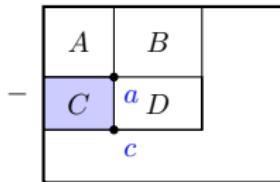
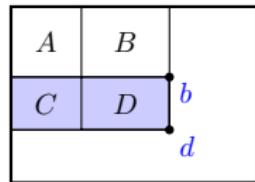
=

- then, the sum over any rectangle ( $D$ ) can be evaluated by 3 scalar operations on its vertices ( $a, b, c, d$ )

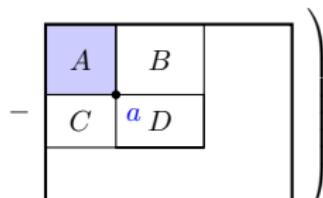
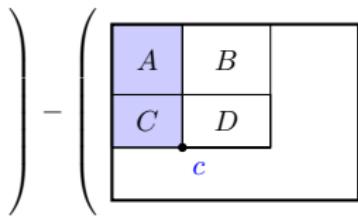
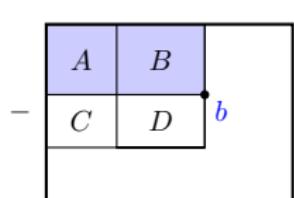
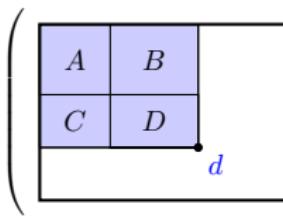
# integral image: use



=



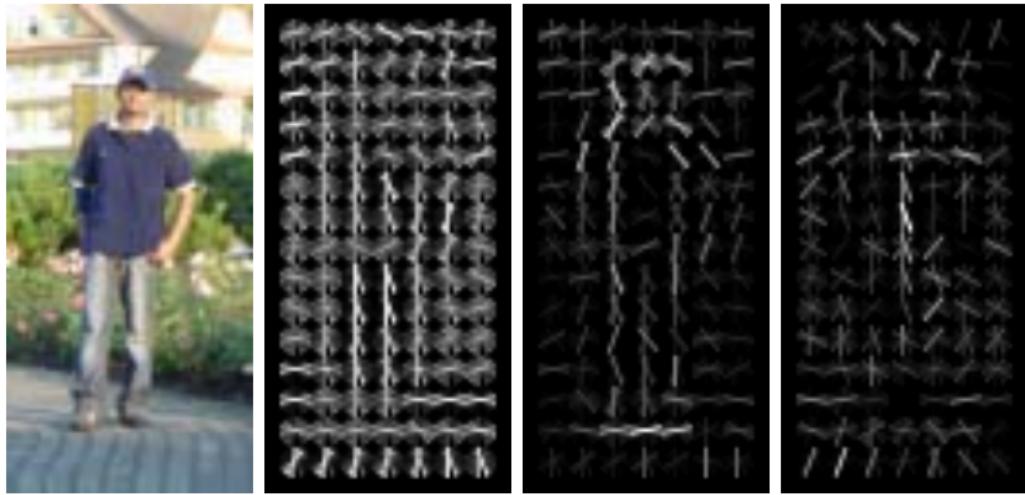
=



- then, the sum over any rectangle ( $D$ ) can be evaluated by 3 scalar operations on its vertices ( $a, b, c, d$ )

# histogram of oriented gradients (HOG)

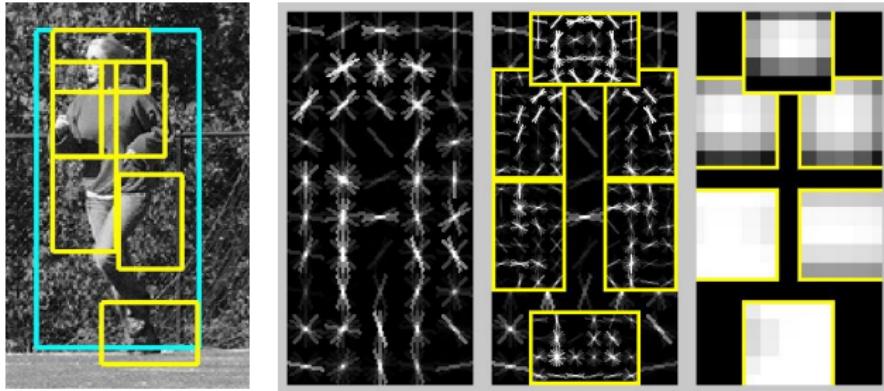
[Dalal and Triggs 2005]



- dense, SIFT-like descriptors
- SVM classifier
- **sliding window** detection at all positions and scales

# deformable part model (DPM)

[Felzenszwalb et al. 2008]

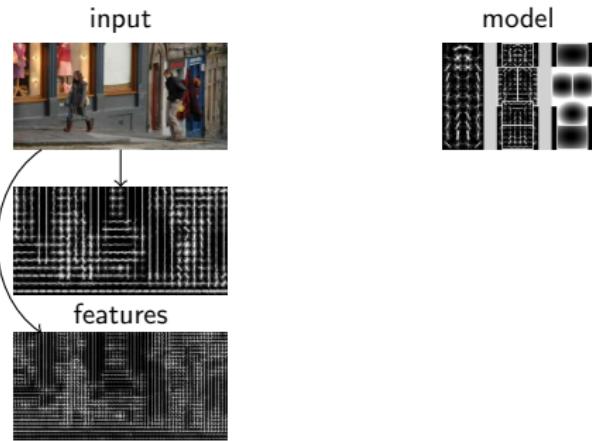


- appearance represented by HOG
- spatial configuration inspired by “pictorial structures”
- part locations treated as latent variables: **latent SVM**

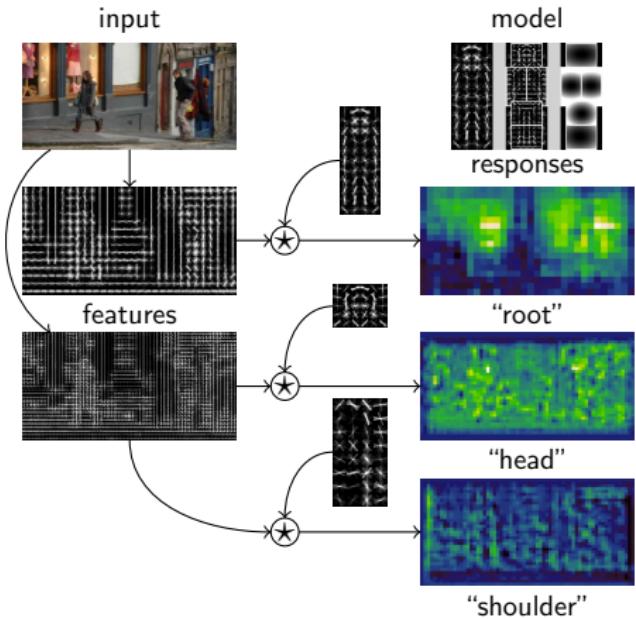
# deformable part model: inference



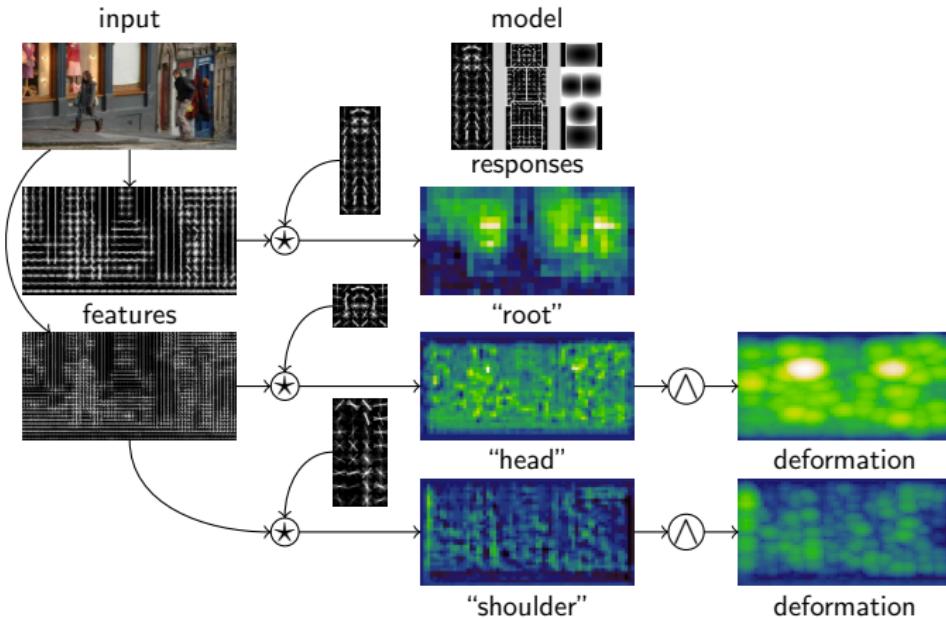
## deformable part model: inference



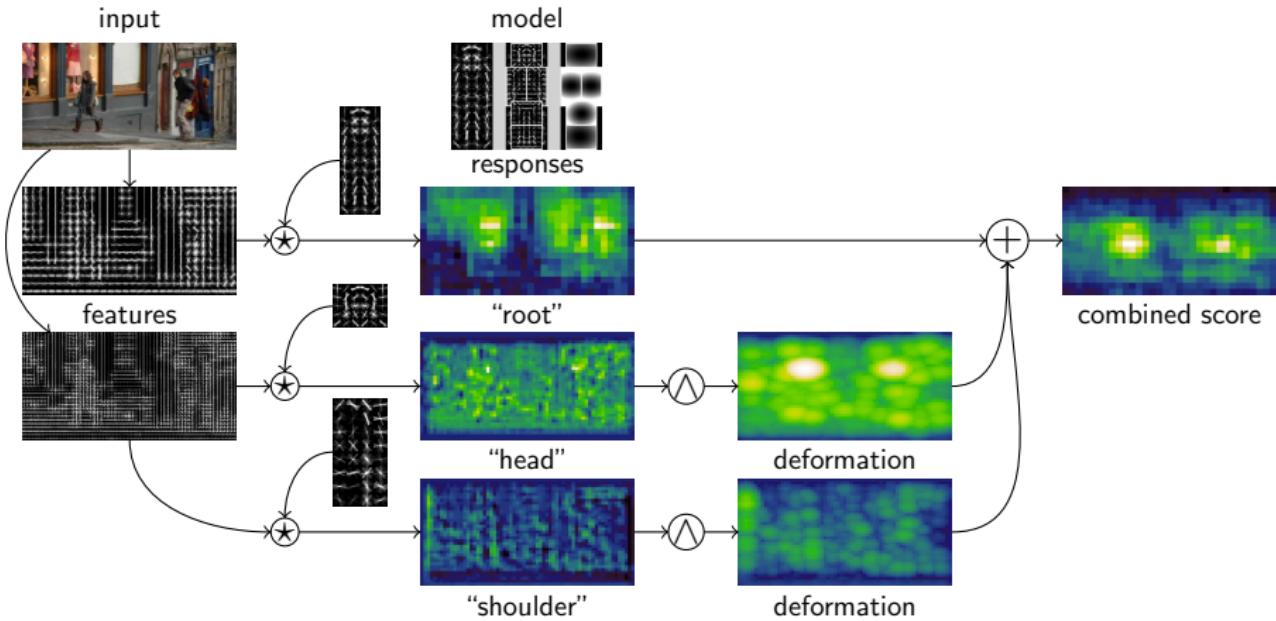
## deformable part model: inference



## deformable part model: inference



## deformable part model: inference



# hard example mining (bootstrapping)

- an example is called **hard** for a model with parameters  $\theta$  if it contributes non-zero loss (is incorrectly classified or inside the margin); otherwise **easy**
- repeat:
  - 1 optimize the model  $\theta$  on a subset  $C$  (**cache**) of the training set  $D$
  - 2 if all hard examples of  $D$  are included in  $C$ , stop
  - 3 **shrink**: remove any number of easy examples from  $C$
  - 4 **grow**: add to  $C$  any number of new samples from  $D$ , including at least a new hard one
- this algorithm terminates and finds the optimal model for  $D$

# hard example mining (bootstrapping)

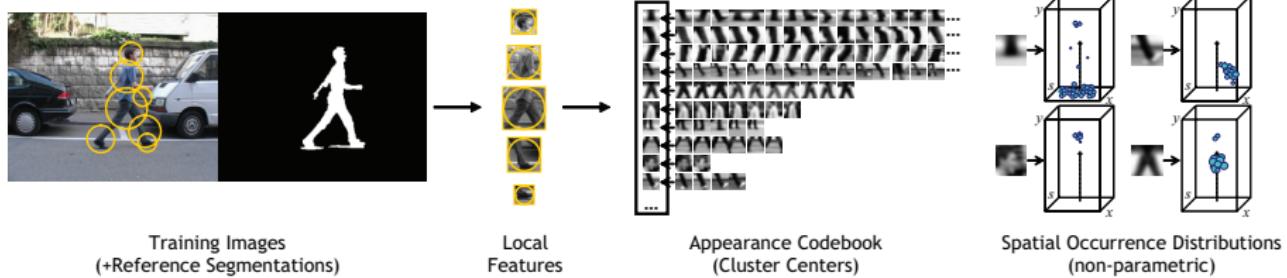
- an example is called **hard** for a model with parameters  $\theta$  if it contributes non-zero loss (is incorrectly classified or inside the margin); otherwise **easy**
- repeat:
  - 1 optimize the model  $\theta$  on a subset  $C$  (**cache**) of the training set  $D$
  - 2 if all hard examples of  $D$  are included in  $C$ , stop
  - 3 **shrink**: remove any number of easy examples from  $C$
  - 4 **grow**: add to  $C$  any number of new samples from  $D$ , including at least a new hard one
- this algorithm terminates and finds the optimal model for  $D$

## hard example mining (bootstrapping)

- an example is called **hard** for a model with parameters  $\theta$  if it contributes non-zero loss (is incorrectly classified or inside the margin); otherwise **easy**
- repeat:
  - 1 optimize the model  $\theta$  on a subset  $C$  (**cache**) of the training set  $D$
  - 2 if all hard examples of  $D$  are included in  $C$ , stop
  - 3 **shrink**: remove any number of easy examples from  $C$
  - 4 **grow**: add to  $C$  any number of new samples from  $D$ , including at least a new hard one
- this algorithm terminates and finds the optimal model for  $D$

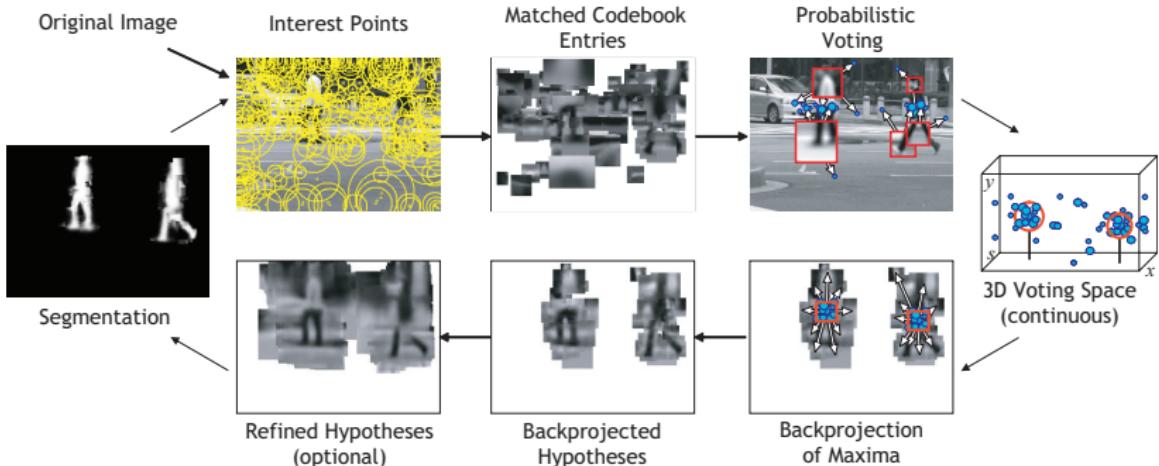
# implicit shape model (ISM): training

[Leibe et al. 2008]



- local features and descriptors extracted on training images
- appearance codebook built
- **spatial occurrence distribution** of features learned, relative to ground truth bounding boxes

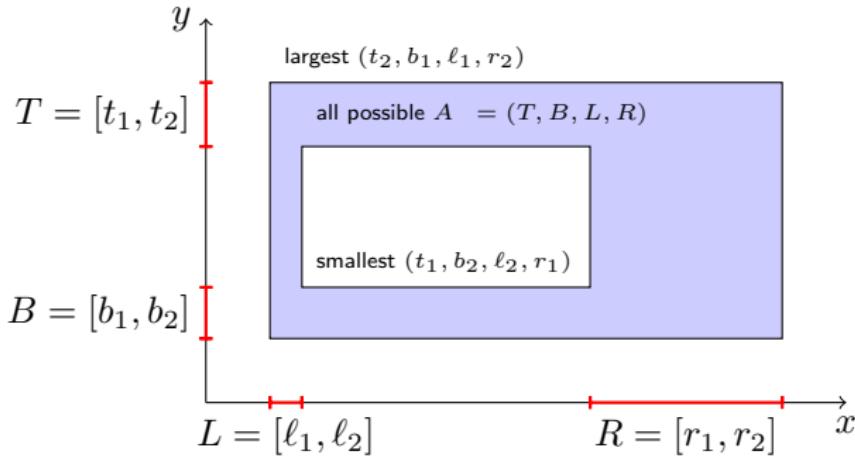
# implicit shape model (ISM): inference



- local features and descriptors extracted on test image
- descriptors assigned to visual words
- **generalized Hough transform**: probabilistic class-specific votes for the object center
- optionally, back-project hypotheses for **top-down segmentation**

# efficient subwindow search (ESS)

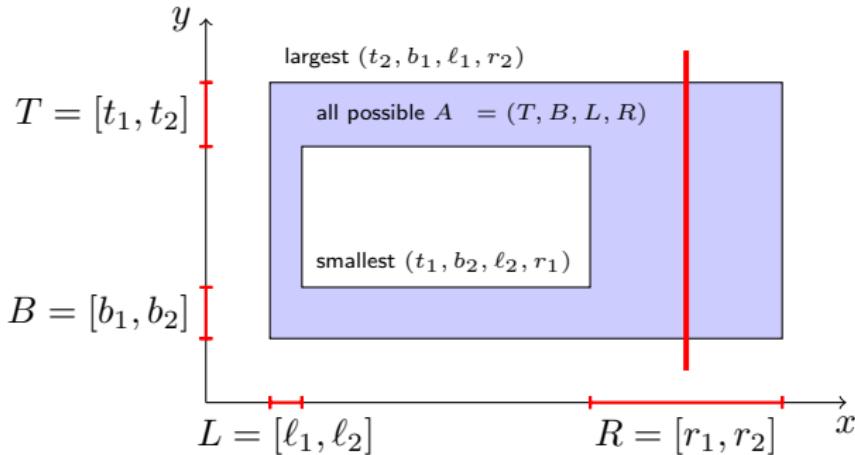
[Lampert et al. 2008]



- the filled area  $A$  represents the set of all rectangles lying in this area
- this set is **split** as  $A = A_1 \cup A_2$  along the largest side and bounds of the objective function are estimated for both subsets
- optimization is performed by **branch-and-bound**

# efficient subwindow search (ESS)

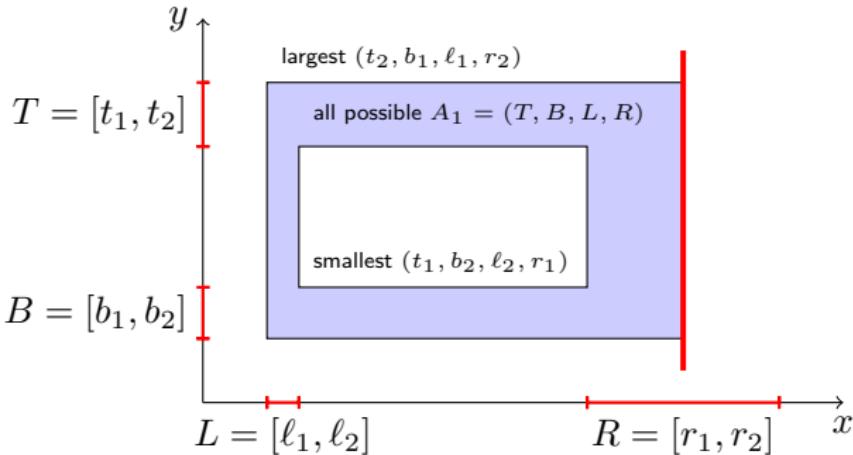
[Lampert et al. 2008]



- the filled area  $A$  represents the set of all rectangles lying in this area
- this set is **split** as  $A = A_1 \cup A_2$  along the largest side and bounds of the objective function are estimated for both subsets
- optimization is performed by **branch-and-bound**

# efficient subwindow search (ESS)

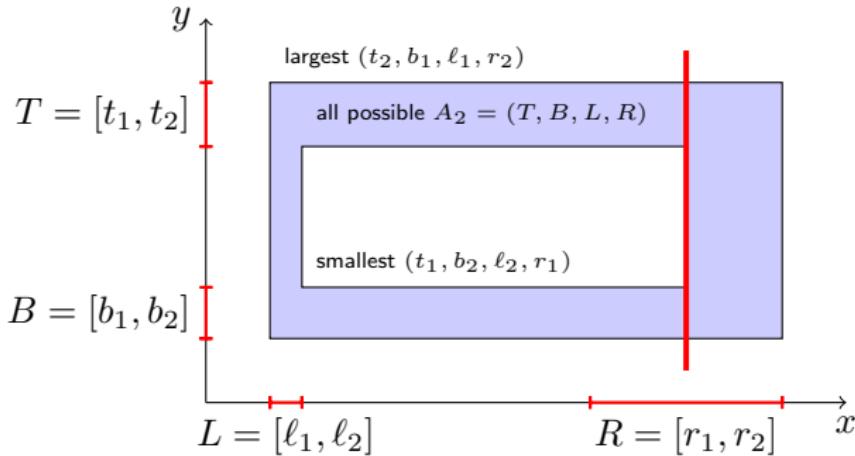
[Lampert et al. 2008]



- the filled area  $A$  represents the set of all rectangles lying in this area
- this set is **split** as  $A = A_1 \cup A_2$  along the largest side and bounds of the objective function are estimated for both subsets
- optimization is performed by **branch-and-bound**

# efficient subwindow search (ESS)

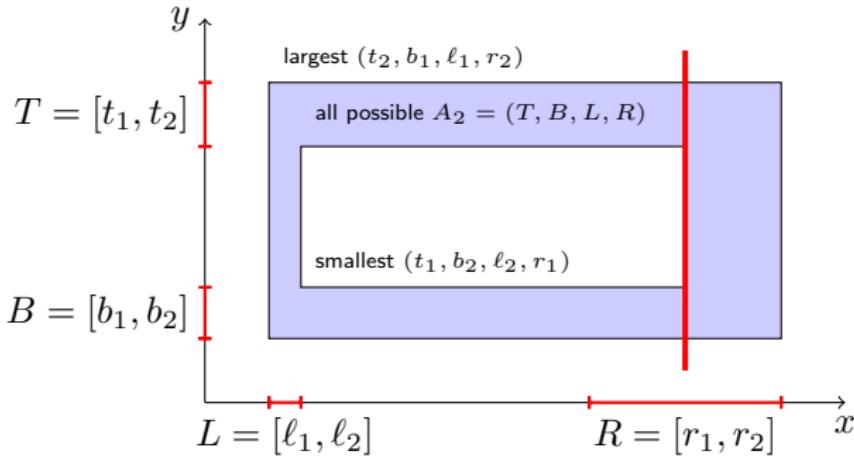
[Lampert et al. 2008]



- the filled area  $A$  represents the set of all rectangles lying in this area
- this set is **split** as  $A = A_1 \cup A_2$  along the largest side and bounds of the objective function are estimated for both subsets
- optimization is performed by **branch-and-bound**

# efficient subwindow search (ESS)

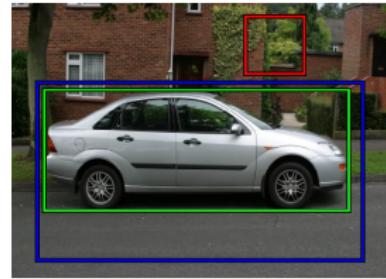
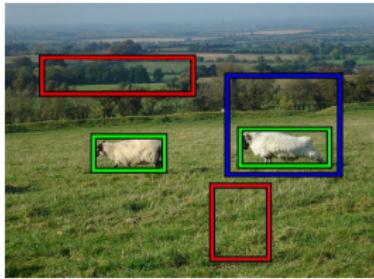
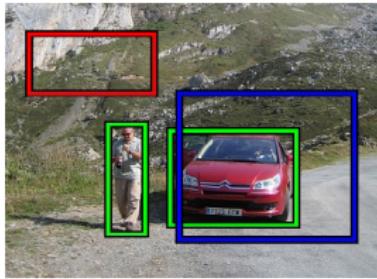
[Lampert et al. 2008]



- the filled area  $A$  represents the set of all rectangles lying in this area
- this set is **split** as  $A = A_1 \cup A_2$  along the largest side and bounds of the objective function are estimated for both subsets
- optimization is performed by **branch-and-bound**

# what is an object?

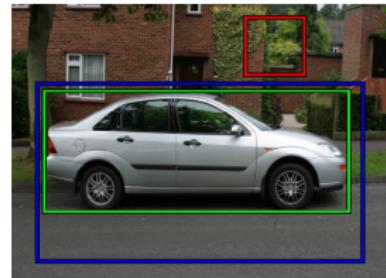
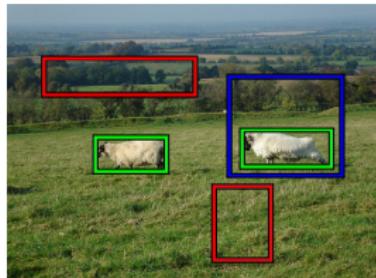
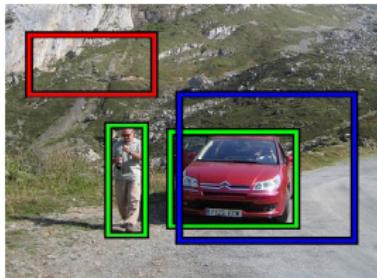
[Alexe et al. 2010]



- seek a generic, **class-agnostic** objectness measure, quantifying how likely it is for an image region to contain an object
- if the measure is simple and fast to compute, it can yield a number of candidate **object proposals** or **regions of interest** (RoI) where to apply a more expensive classifier
- score the **blue** regions, partially covering the objects, lower than the **green** ground truth regions
- even lower the **red** regions containing only stuff or small object parts

# what is an object?

[Alexe et al. 2010]



- seek a generic, **class-agnostic** objectness measure, quantifying how likely it is for an image region to contain an object
- if the measure is simple and fast to compute, it can yield a number of candidate **object proposals** or **regions of interest** (RoI) where to apply a more expensive classifier
- score the **blue** regions, partially covering the objects, lower than the **green** ground truth regions
- even lower the **red** regions containing only stuff or small object parts

# selective search (SS)

[van de Sande et al. 2011]



input image



ground truth

# selective search (SS)

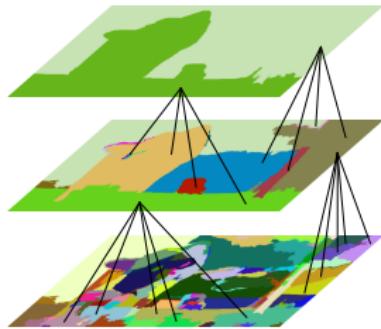
[van de Sande et al. 2011]



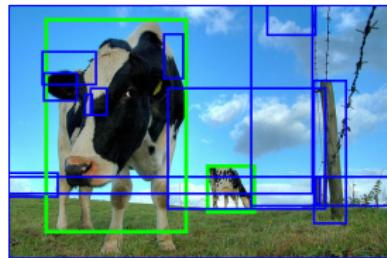
input image



ground truth



hierarchical grouping



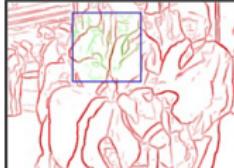
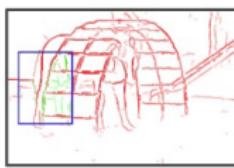
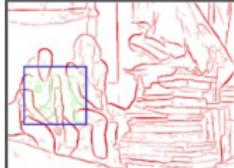
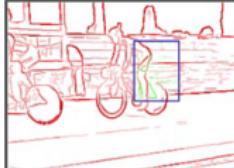
object proposals

# selective search (SS)

- hierarchical segmentation at all scales
- simple geometric and appearance features (e.g. size, texture)
- high recall:  $\sim 97\%$  of ground truth objects found with  $\sim 1000 - 2000$  proposals/image at  $\sim 2-5\text{s}/\text{image}$

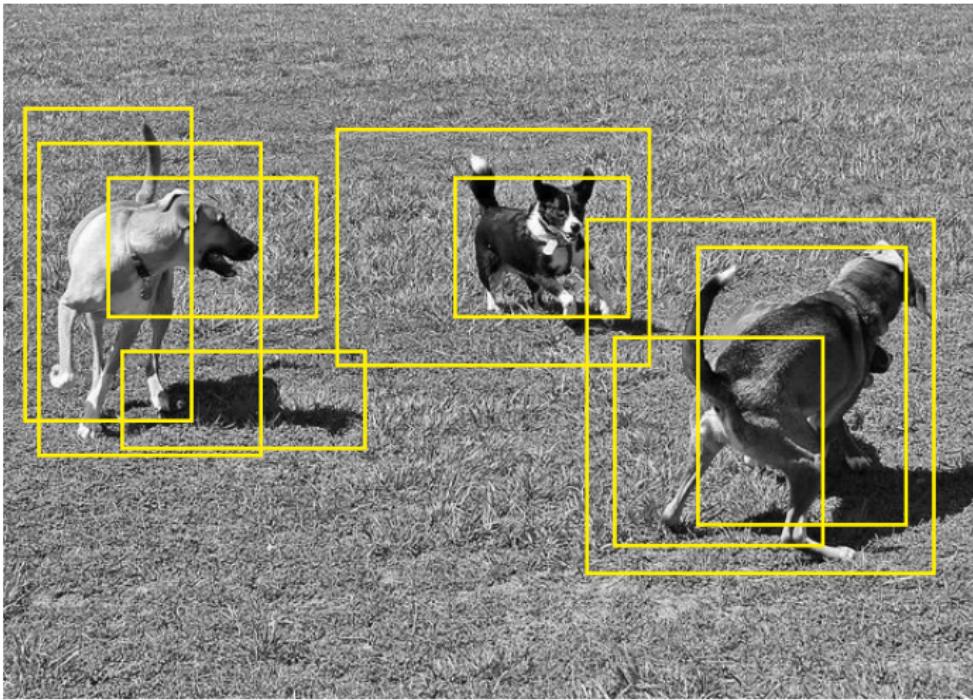
# edge boxes (EB)

[Zitnick and Dollar 2014]

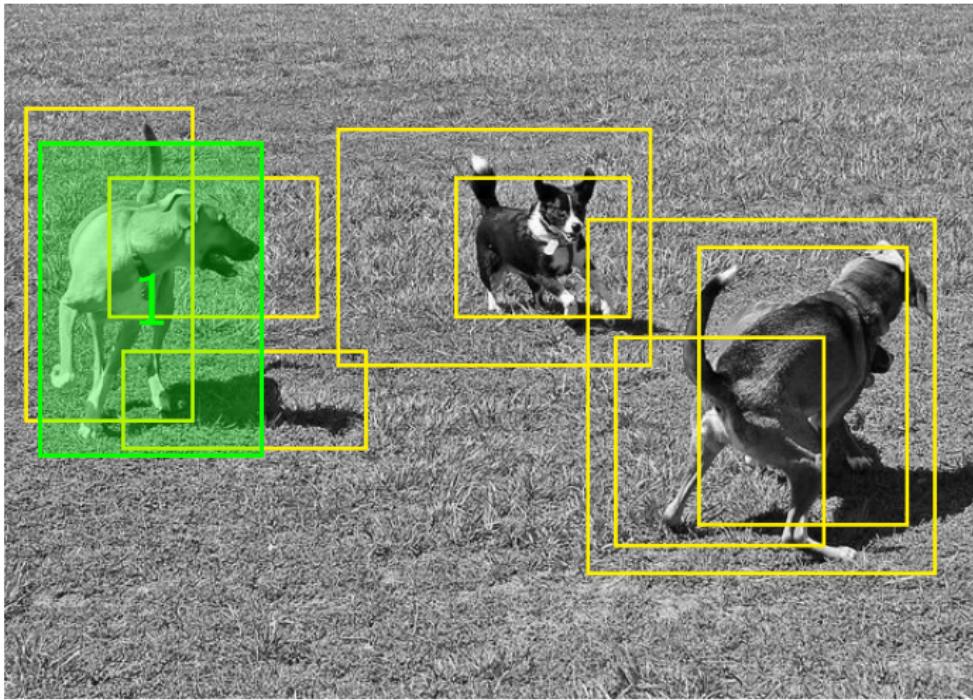


- fast evaluation of millions of regions of different scales/aspect ratios at different positions
- measures edges that are contained in a region and do not intersect its boundary
- performance similar to SS, but at  $\sim 0.25\text{s}/\text{image}$  on average

# non-maximum suppression (NMS)

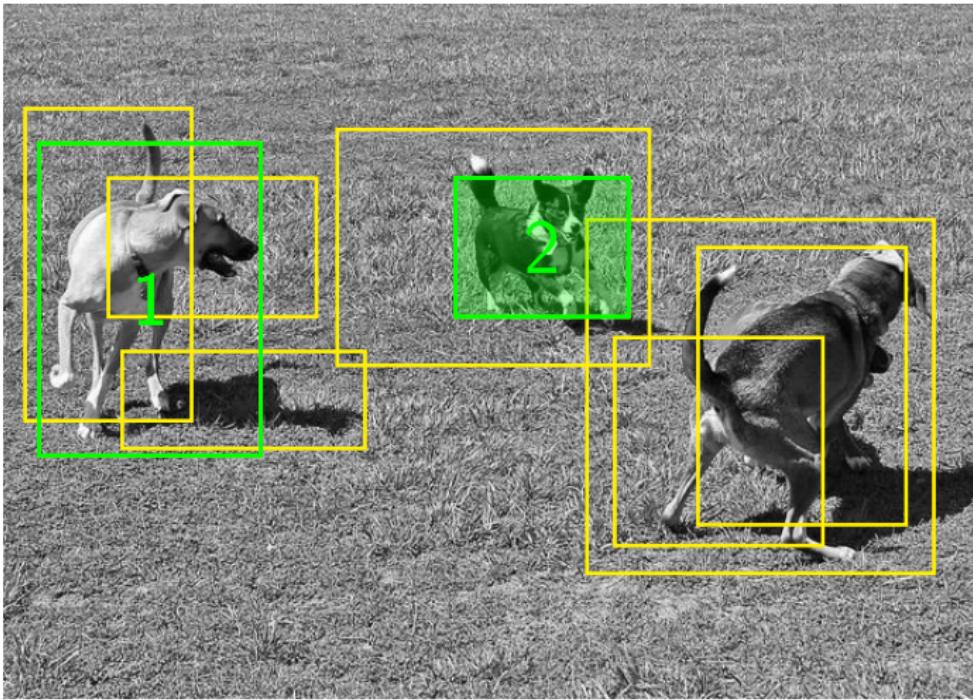


# non-maximum suppression (NMS)



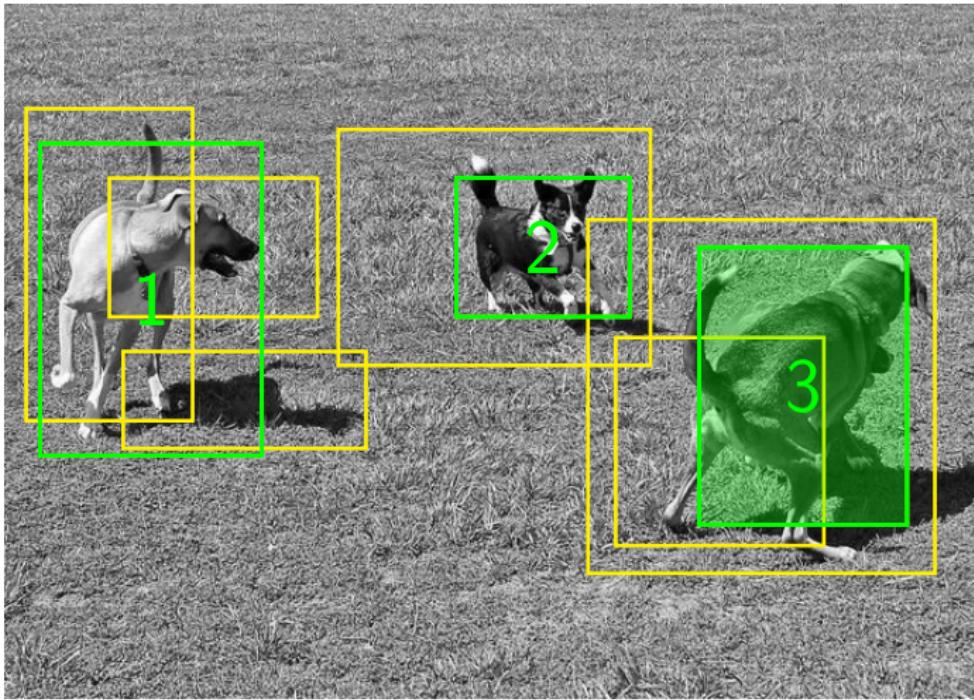
region 1 remains

# non-maximum suppression (NMS)



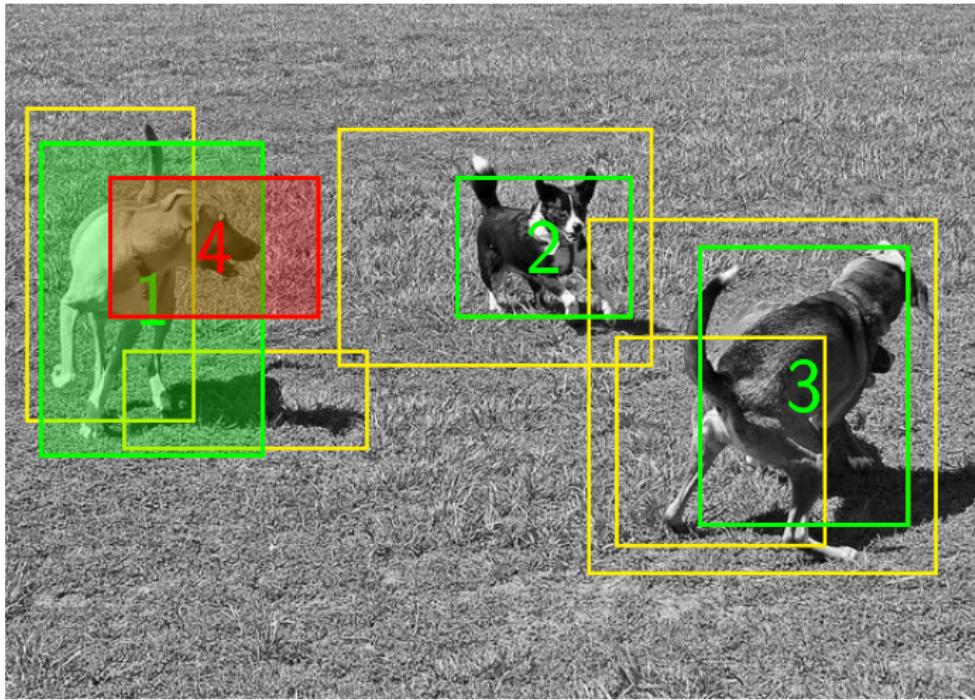
region 2 remains

# non-maximum suppression (NMS)



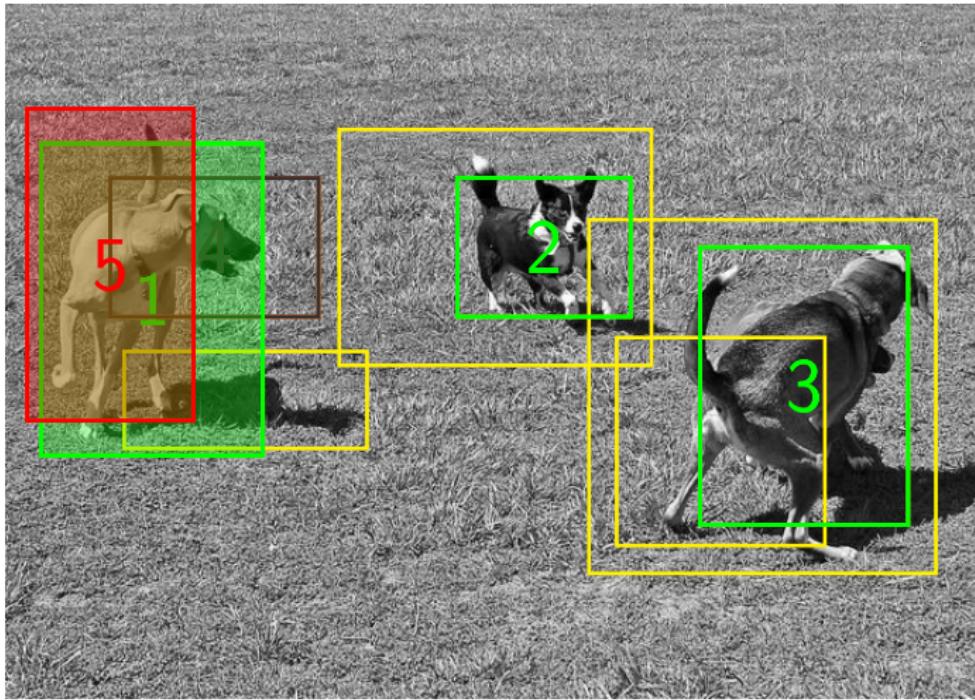
region 3 remains

## non-maximum suppression (NMS)



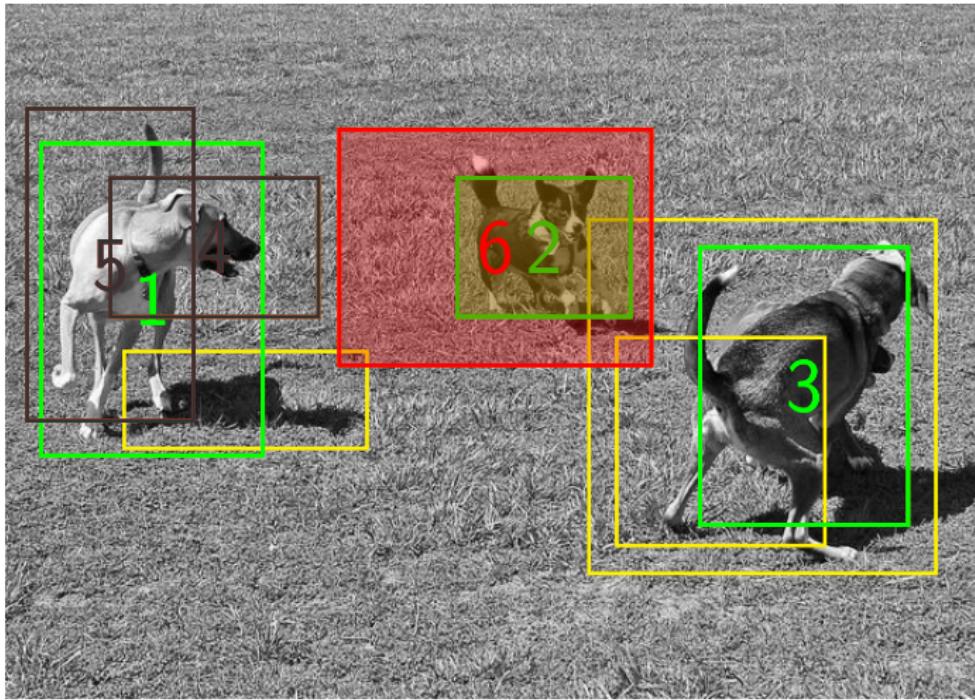
region 4 is rejected because  $J(r_4, r_1) = 0.2750 > 0.25$

## non-maximum suppression (NMS)



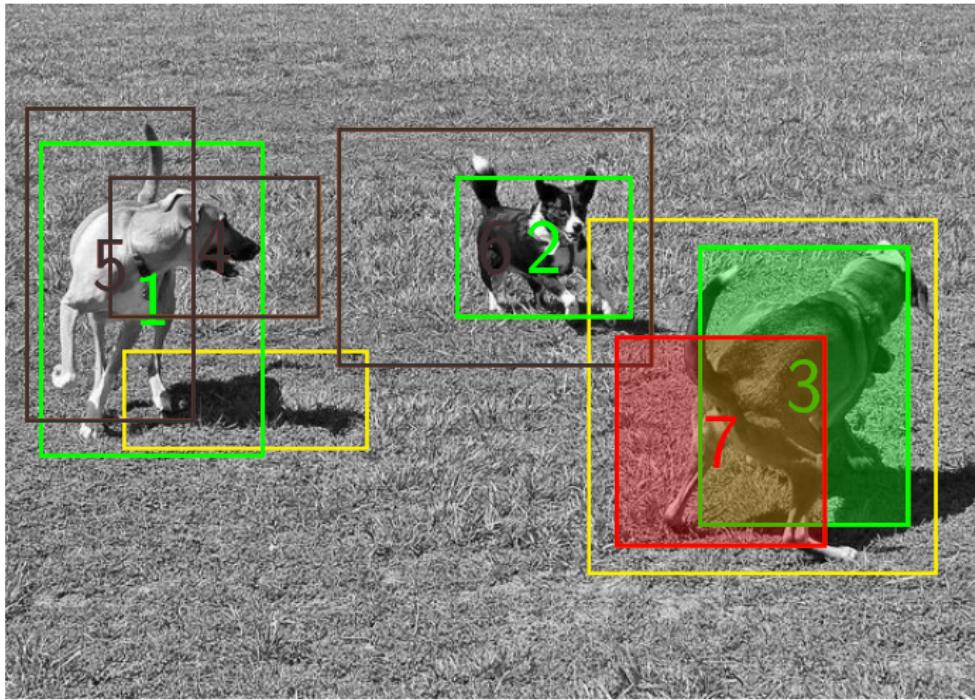
region 5 is rejected because  $J(r_5, r_1) = 0.5366 > 0.25$

## non-maximum suppression (NMS)



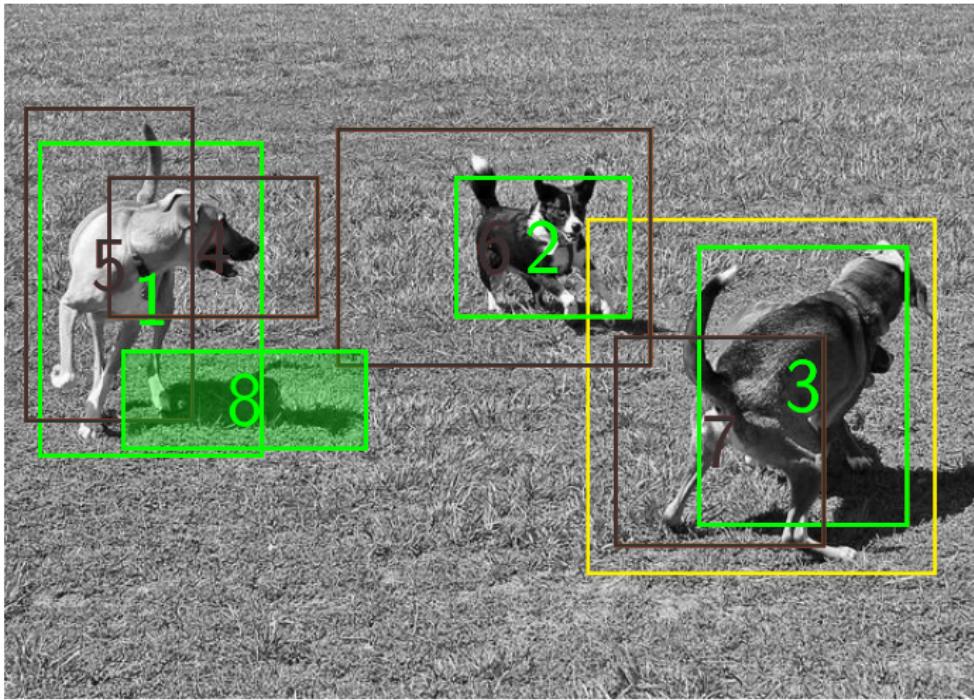
region 6 is rejected because  $J(r_6, r_2) = 0.3268 > 0.25$

## non-maximum suppression (NMS)



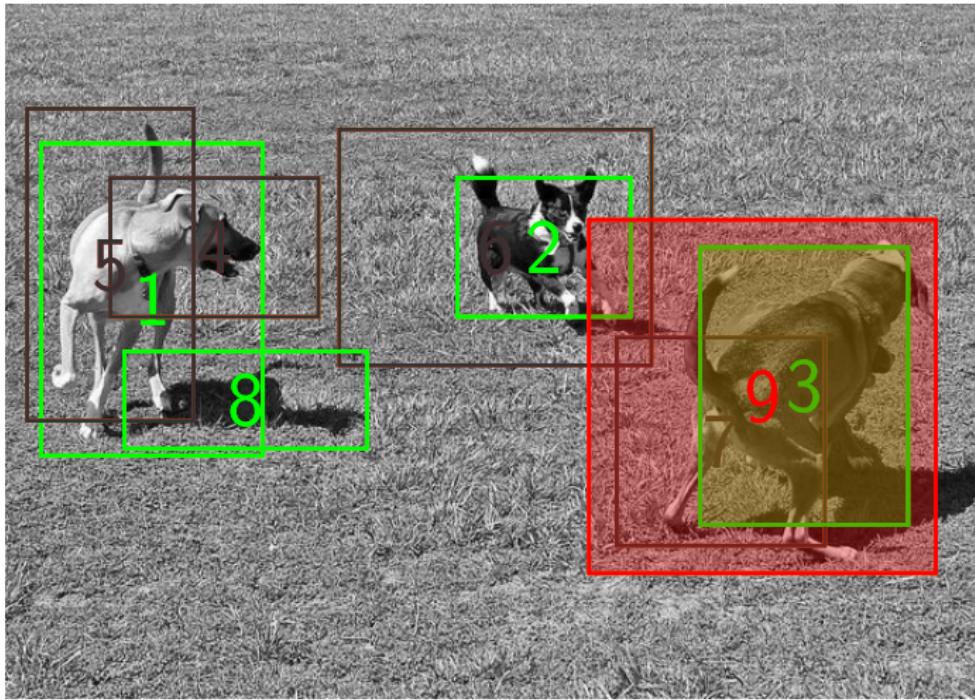
region 7 is rejected because  $J(r_7, r_3) = 0.3011 > 0.25$

# non-maximum suppression (NMS)



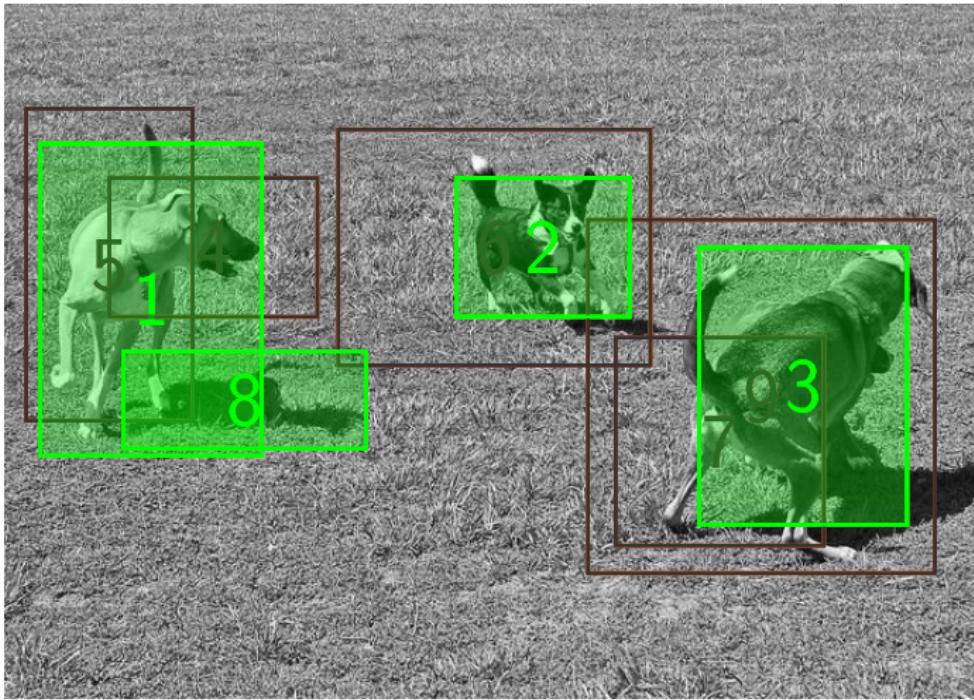
region 8 remains

## non-maximum suppression (NMS)



region 9 is rejected because  $J(r_9, r_3) = 0.4706 > 0.25$

# non-maximum suppression (NMS)



in the end, regions 1, 2, 3, 8 remain

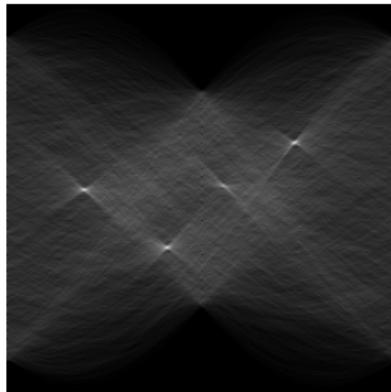
## non-maximum suppression on regions

- given regions  $r_1, r_2, \dots$  of each class independently, ranked by decreasing order of confidence score
- for  $i = 2, 3, \dots$ , reject region  $r_i$  if it has intersection-over-union (IoU) overlap higher than a threshold  $\tau$

$$J(r_i, r_j) > \tau$$

with some higher scoring region  $r_j$  with  $j < i$  that has not been rejected

## non-maximum suppression is everywhere



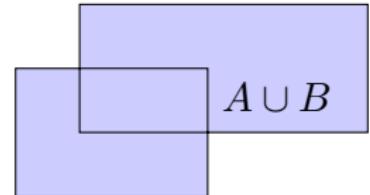
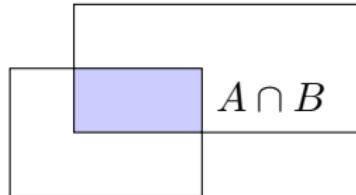
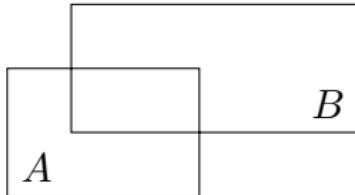
accumulator



local maxima

- we have used NMS to reject **pixels** or 1d-vector elements (rather than regions) according to some neighborhood relation, in
  - corner detection
  - feature point tracking
  - SIFT dominant orientation selection
  - Hough transform

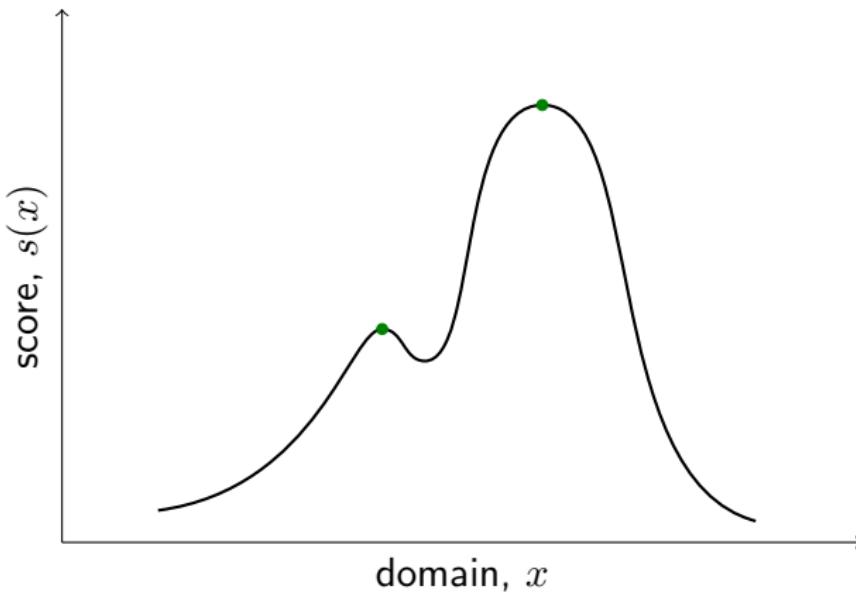
# region overlap



- given regions  $A, B \subset \mathbb{R}^2$  represented as planar point sets (including interior)
- their **intersection over union** (IoU) or **Jaccard index** is

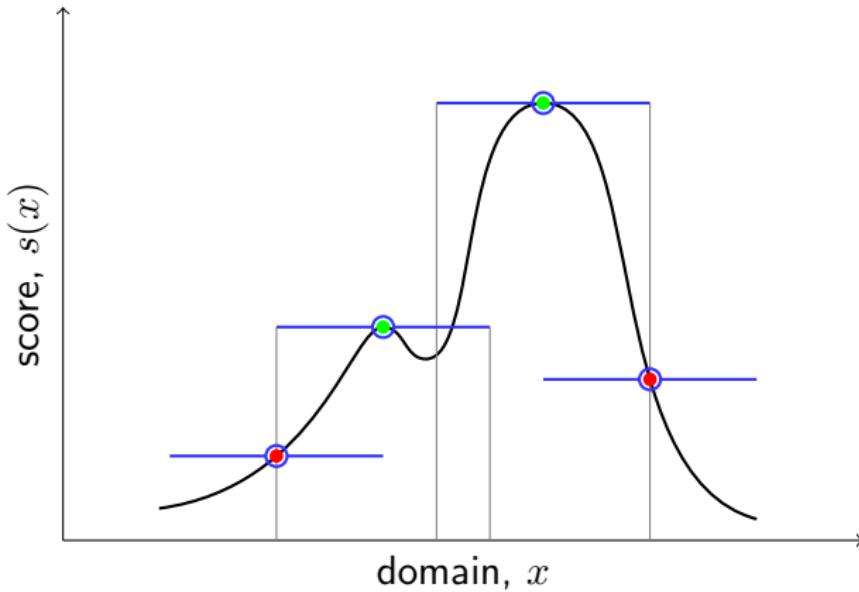
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

# the problem of non-maximum suppression



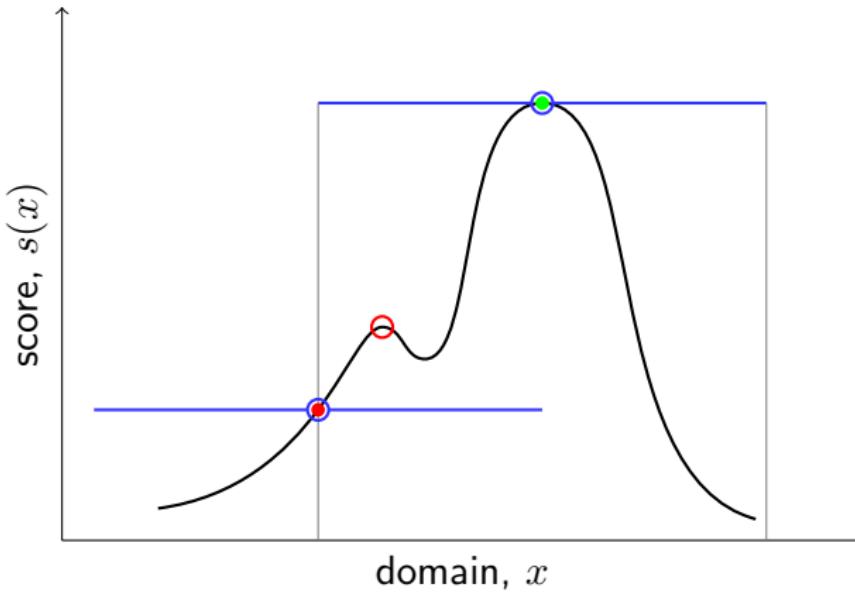
- ground truth positions

# the problem of non-maximum suppression



- with a narrow neighborhood, there are two true positives (•) but also two false positives (●): **precision is low**

## the problem of non-maximum suppression



- with a wide neighborhood, there is only one true positive (•), one false positive (●) and one false negative (○): **recall is low**

## non-maximum suppression

- there are several recent attempts to improve NMS, e.g. merging or down-weighting instead of rejecting, replace it by a CNN, or integrate a differentiable version so that the entire pipeline is **end-to-end trainable**
- here we assume there is **always** NMS as the last post-processing stage after each detector

# detection evaluation

[Russakovsky et al. 2015]

- for each image and for each class independently, rank predicted regions by descending order of confidence and assign each region  $r$  to the ground truth region  $g^* = \arg \max_g J(r, g)$  of maximum overlap if  $J(r, g^*) > \tau$  and mark it as true positive, else false
- each ground truth region can be assigned up to one predicted region
- now for each class independently, rank predicted regions of all images by descending order of confidence and compute average precision (AP) according to true/false labels
- the mean average precision (mAP) is the mean over classes

# detection evaluation

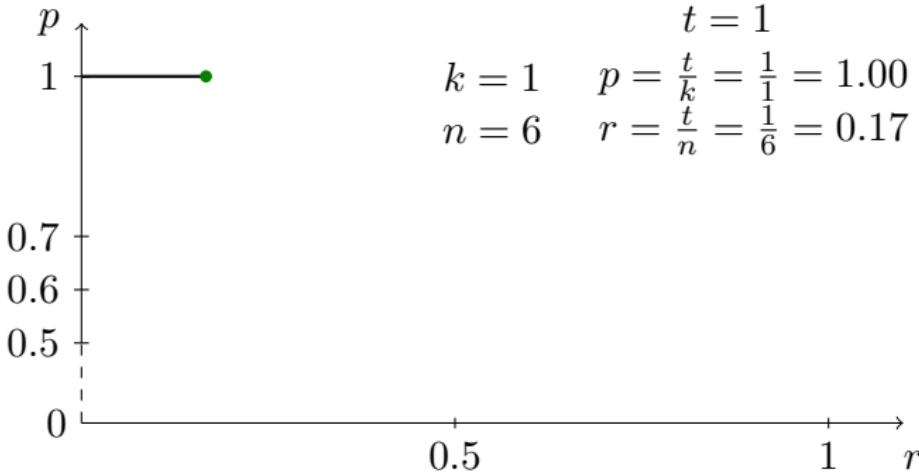
[Russakovsky et al. 2015]

- for **each image** and for **each class** independently, rank predicted regions by descending order of confidence and assign each region  $r$  to the ground truth region  $g^* = \arg \max_g J(r, g)$  of **maximum overlap** if  $J(r, g^*) > \tau$  and mark it as **true** positive, else **false**
- each ground truth region can be assigned up to one predicted region
- now for **each class** independently, rank predicted regions of **all images** by descending order of confidence and compute **average precision** (AP) according to true/false labels
- the **mean average precision** (mAP) is the mean over classes

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

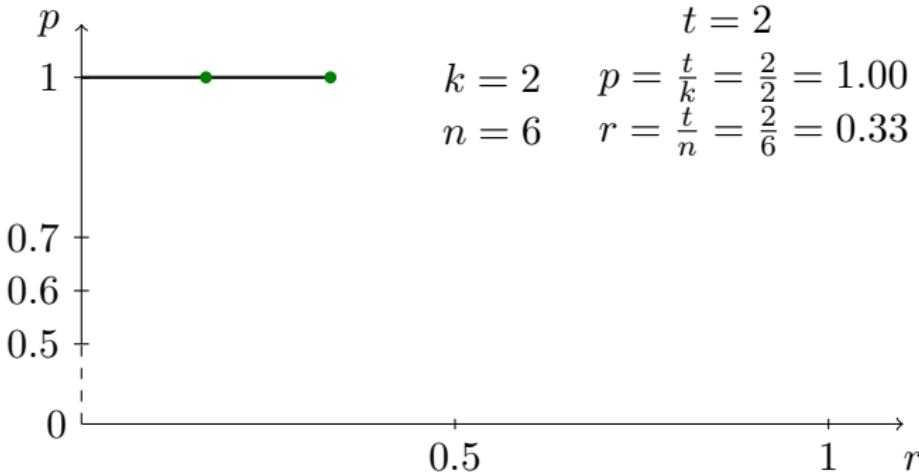


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

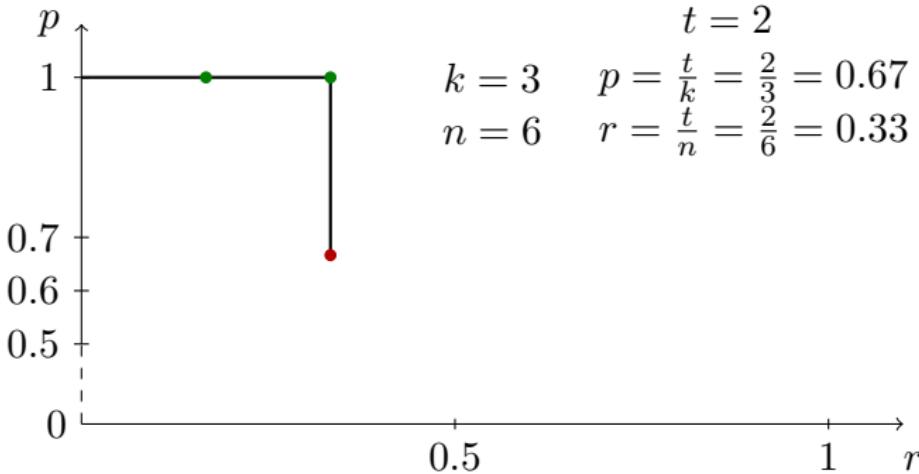


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

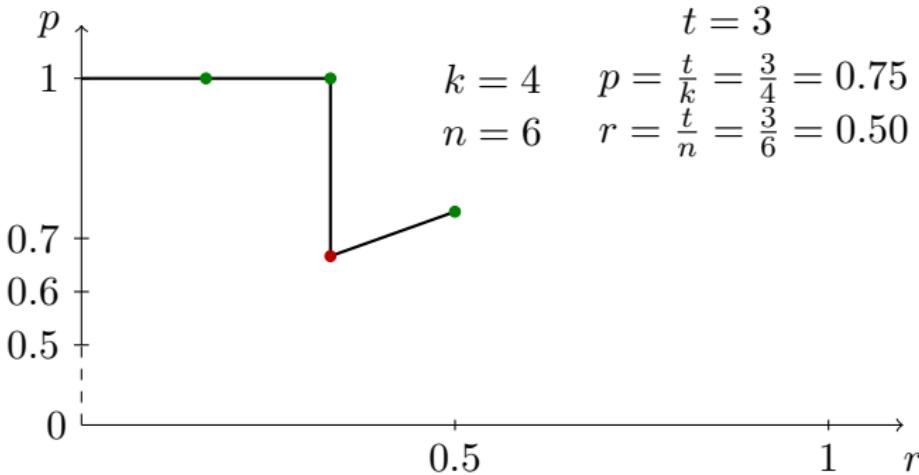


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

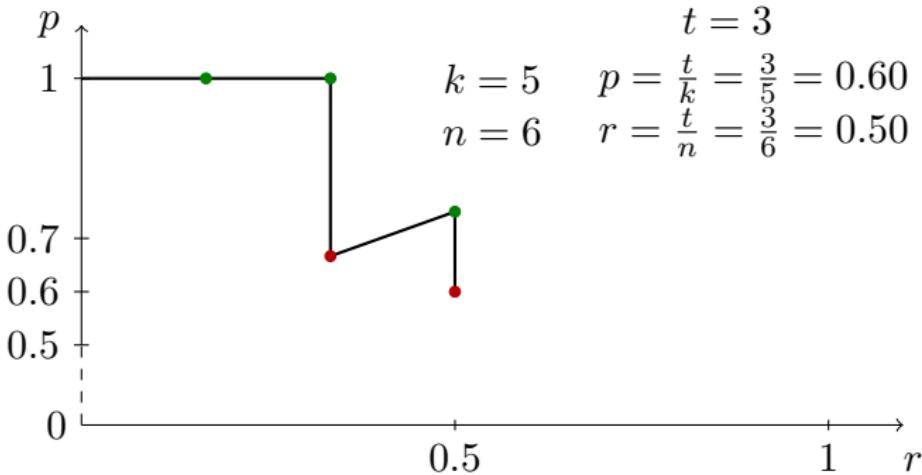


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

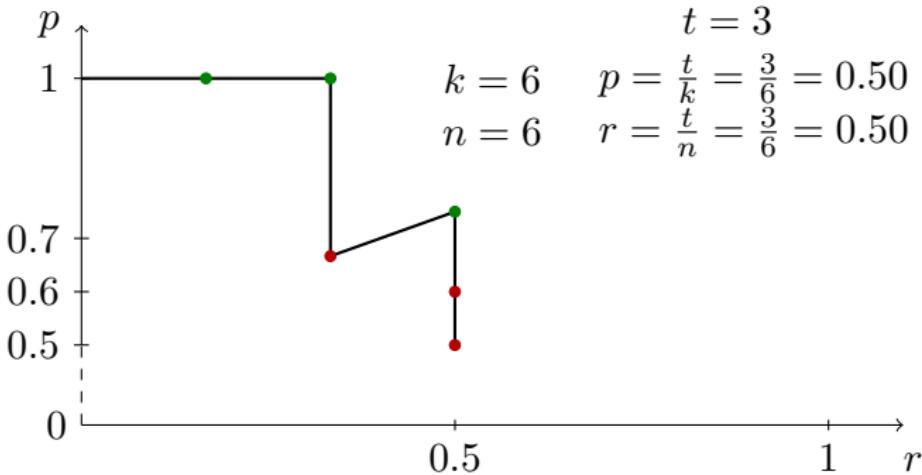


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

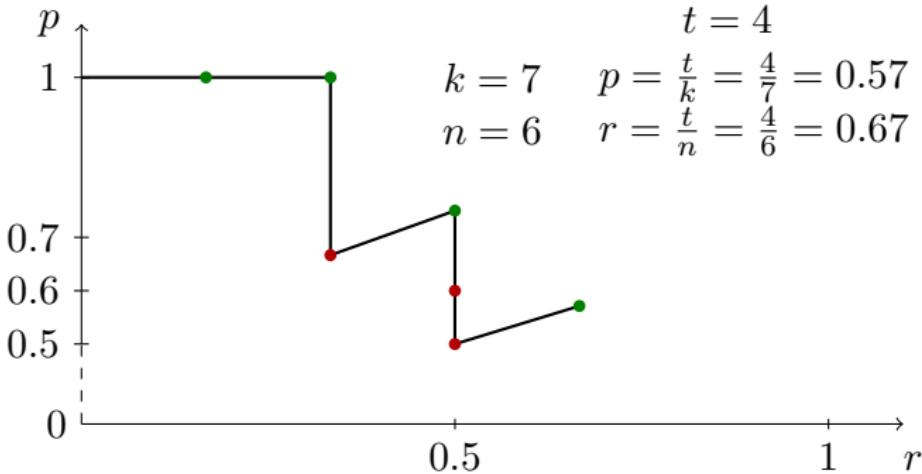


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

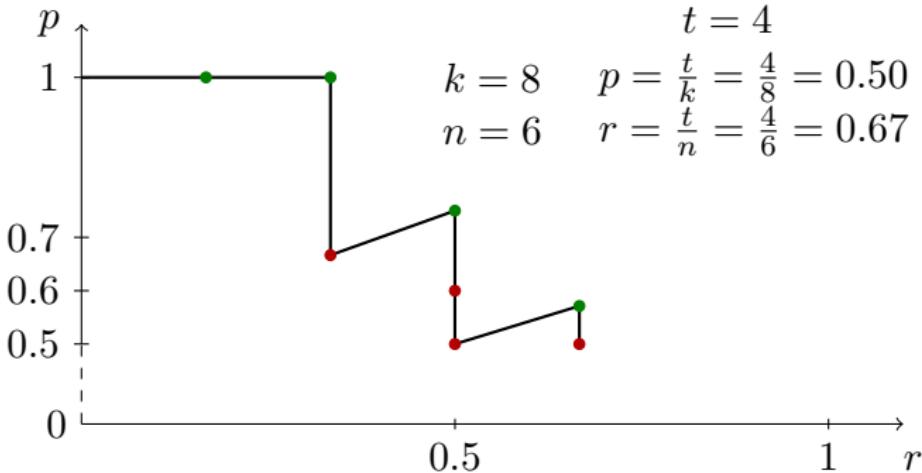


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

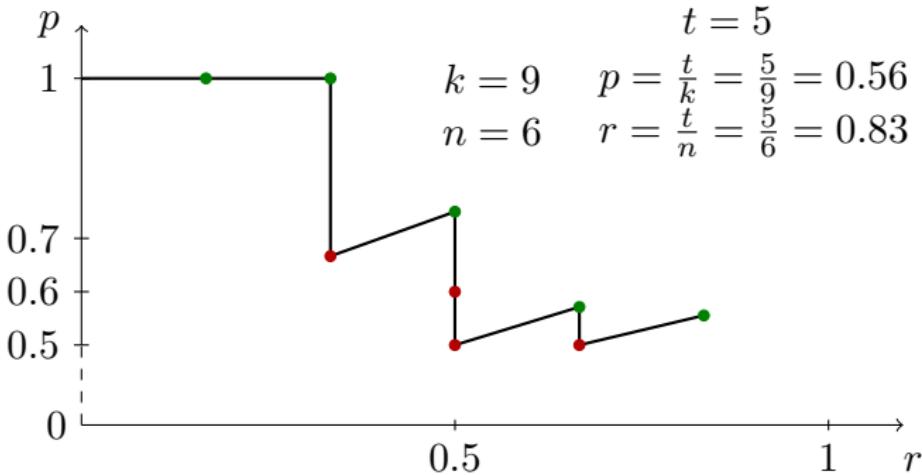


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

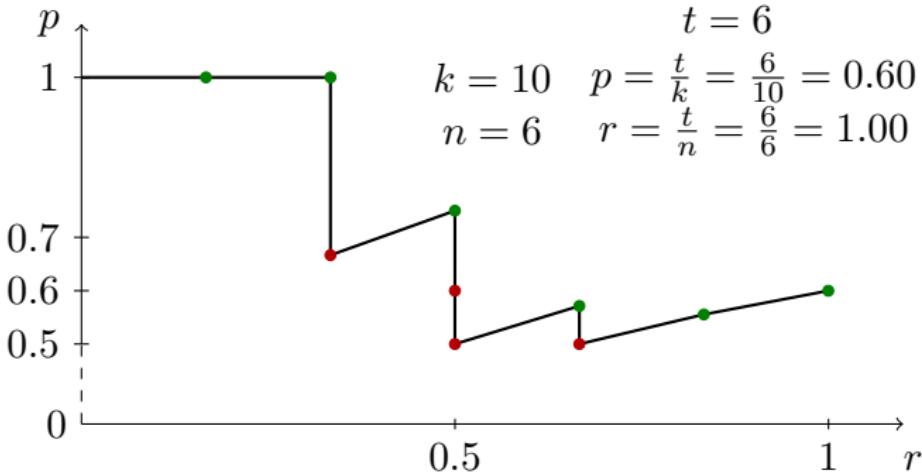


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

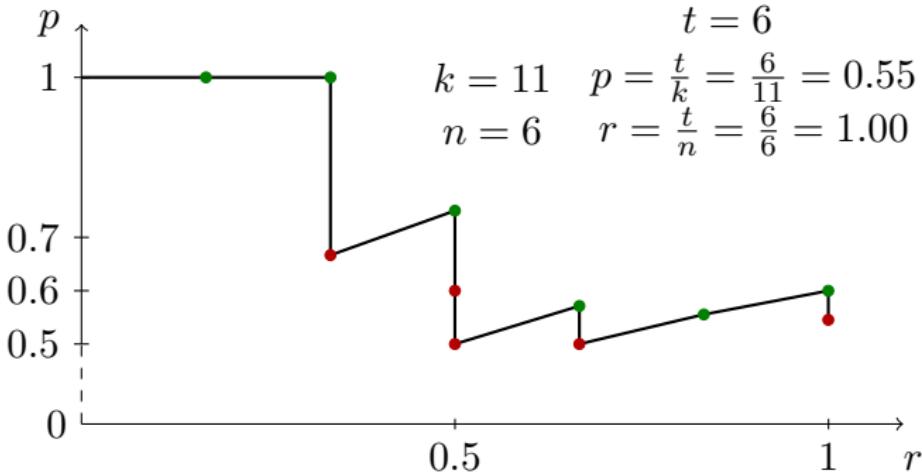


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

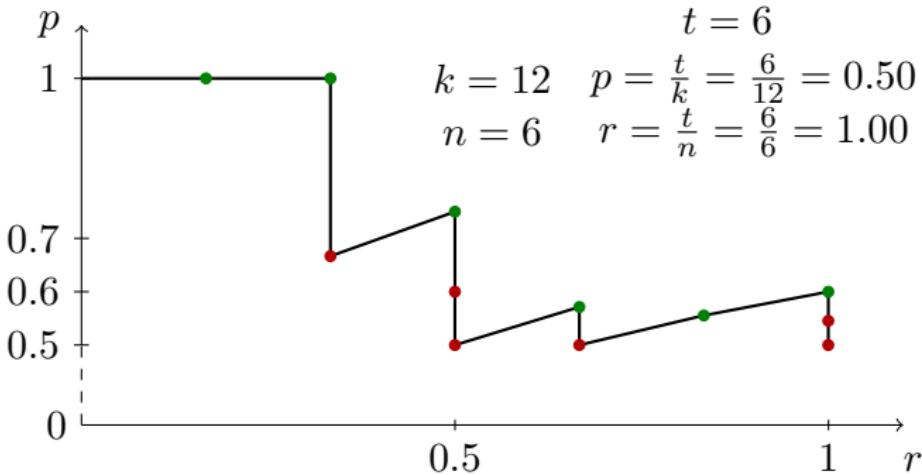


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F

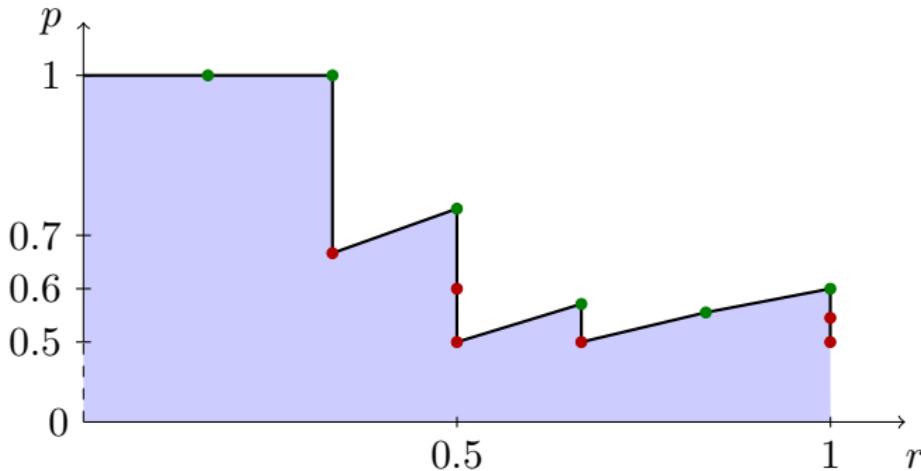


- # total ground truth  $n$ , current rank  $k$ , # true positives  $t$
- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F



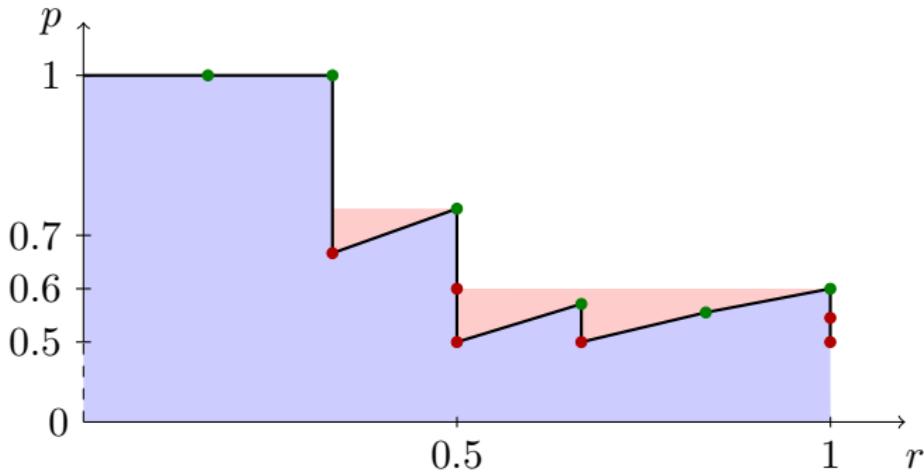
- average precision = area under curve

- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# average precision (AP)

- ranked list of items with true/false labels

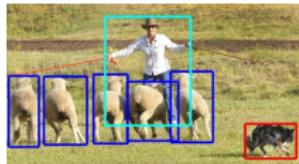
1	2	3	4	5	6	7	8	9	10	11	12
T	T	F	T	F	F	T	F	T	T	F	F



- average precision = area under curve (filled-in curve)

- precision  $p = \frac{t}{k}$ , recall  $r = \frac{t}{n}$

# object detection datasets



- **PASCAL** VOC 2007-12: 20 classes; images 5-11k train/val, 5-11k test (public for 2007)
- **ImageNet** ILSVRC 2010-17: 200 classes (subset or merged from classification task); images 400-450k train (partially annotated), 20k val, 40k test
- **COCO** 2015-: 80 classes; images 80k train, 40k val (115k/5k in 2017), 40k test, 120k unlabeled; smaller objects
- **Open Images** 2018-: 600 classes; images 1.74M train, 41k val, 125k test

Everingham *et al.* IJCV 2015. The PASCAL Visual Object Classes Challenge: a Retrospective.

Russakovsky *et al.* IJCV 2015. Imagenet Large Scale Visual Recognition Challenge.

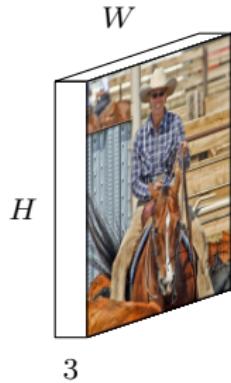
Lin *et al.* ECCV 2014. Microsoft COCO: Common Objects in Context.

Kuznetsova *et al.* 2018. The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale.

# two-stage detection

# regions with CNN features (R-CNN)

[Girshick et al. 2014]

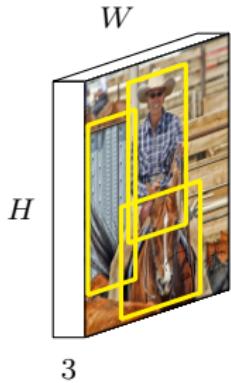


- 3-channel RGB input, fixed width  $W = 500$  pixels
- $\sim 2000$  SS region proposals warped into fixed  $w \times h = 227 \times 227$
- each proposal yields a  $k = 4096$  dimensional feature by CaffeNet
- each feature is classified into  $c$  classes by  $c$  one-vs.-rest SVMs and localized by bounding box regression

Girshick, Donahue, Darrell and Malik. CVPR 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.

# regions with CNN features (R-CNN)

[Girshick et al. 2014]

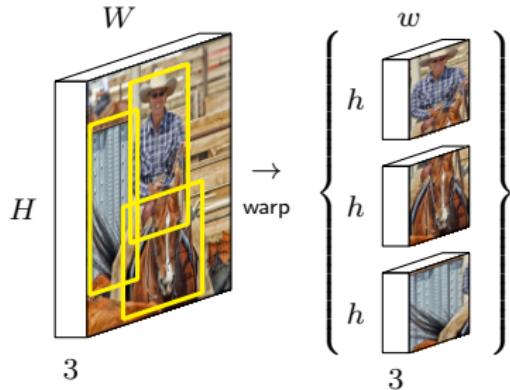


- 3-channel RGB input, fixed width  $W = 500$  pixels
- $\sim 2000$  SS region proposals warped into **fixed**  $w \times h = 227 \times 227$
- each proposal yields a  $k = 4096$  dimensional feature by CaffeNet
- each feature is classified into  $c$  classes by  $c$  one-vs.-rest SVMs and localized by bounding box regression

Girshick, Donahue, Darrell and Malik. CVPR 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation.

# regions with CNN features (R-CNN)

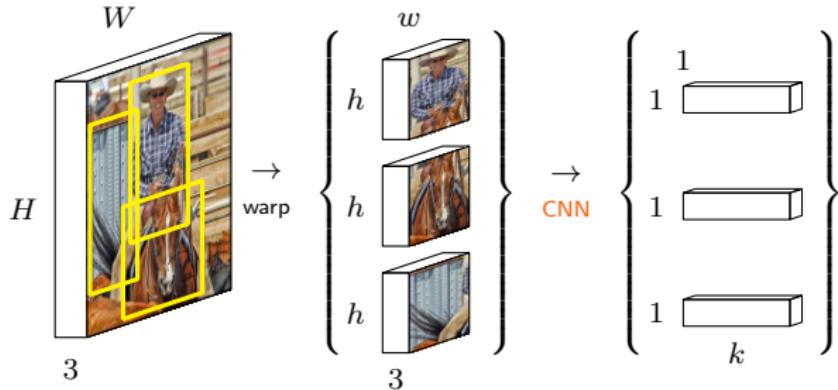
[Girshick et al. 2014]



- 3-channel RGB input, fixed width  $W = 500$  pixels
- $\sim 2000$  SS region proposals warped into **fixed**  $w \times h = 227 \times 227$
- each proposal yields a  $k = 4096$  dimensional feature by CaffeNet
- each feature is classified into  $c$  classes by  $c$  one-vs.-rest SVMs and localized by bounding box regression

# regions with CNN features (R-CNN)

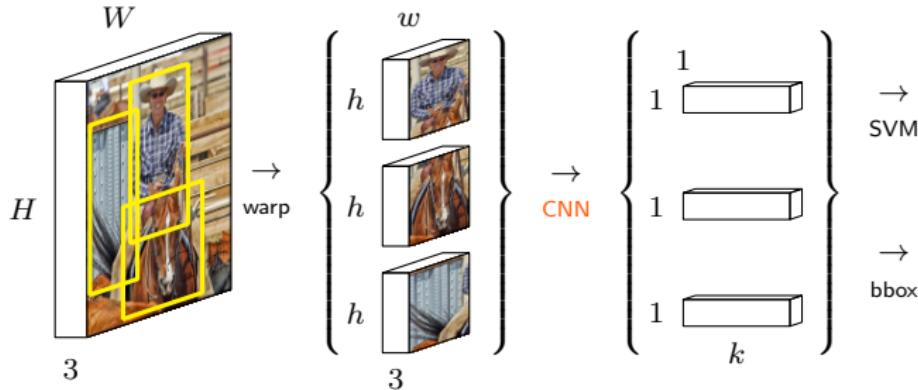
[Girshick et al. 2014]



- 3-channel RGB input, fixed width  $W = 500$  pixels
- $\sim 2000$  SS region proposals warped into **fixed**  $w \times h = 227 \times 227$
- **each proposal** yields a  $k = 4096$  dimensional feature by CaffeNet
- each feature is classified into  $c$  classes by  $c$  one-vs. -rest SVMs and localized by bounding box regression

# regions with CNN features (R-CNN)

[Girshick et al. 2014]



- 3-channel RGB input, fixed width  $W = 500$  pixels
- $\sim 2000$  SS region proposals warped into **fixed**  $w \times h = 227 \times 227$
- **each proposal** yields a  $k = 4096$  dimensional feature by CaffeNet
- each feature is classified into  $c$  classes by  $c$  one-vs. -rest SVMs and localized by bounding box regression

# regions with CNN features (R-CNN)

## pros

- region proposals, SVM classifier and NMS are standard; here one just replaces the features (e.g. HOG) by CNN
- CNN features are 4k-dimensional, compared e.g. to 360k dimensions of previous state of the art
- **transfer learning**: network pre-trained on 1.2M ImageNet images, then ImageNet-specific 1000-way classification layer replaced by randomly initialized  $(c + 1)$ -way ( $c$  classes plus background) and **fine-tuning**

## cons

- **slow** (13s/image): image warped and forwarded through network for each of the  $\sim 2000$  region proposals
- **4 stages**: region extraction, CNN features, SVM classifier, regressor
- positives/negatives defined differently in fine-tuning vs. SVM

# regions with CNN features (R-CNN)

## pros

- region proposals, SVM classifier and NMS are standard; here one just replaces the features (e.g. HOG) by CNN
- CNN features are 4k-dimensional, compared e.g. to 360k dimensions of previous state of the art
- *transfer learning*: network pre-trained on 1.2M ImageNet images, then ImageNet-specific 1000-way classification layer replaced by randomly initialized  $(c + 1)$ -way ( $c$  classes plus background) and *fine-tuning*

## cons

- *slow* (13s/image): image warped and forwarded through network for each of the  $\sim 2000$  region proposals
- *4 stages*: region extraction, CNN features, SVM classifier, regressor
- positives/negatives defined differently in fine-tuning vs. SVM

# regions with CNN features (R-CNN)

## pros

- region proposals, SVM classifier and NMS are standard; here one just replaces the features (e.g. HOG) by CNN
- CNN features are 4k-dimensional, compared e.g. to 360k dimensions of previous state of the art
- **transfer learning**: network pre-trained on 1.2M ImageNet images, then ImageNet-specific 1000-way classification layer replaced by randomly initialized  $(c + 1)$ -way ( $c$  classes plus background) and **fine-tuning**

## cons

- **slow** (13s/image): image warped and forwarded through network for each of the  $\sim 2000$  region proposals
- **4 stages**: region extraction, CNN features, SVM classifier, regressor
- positives/negatives defined differently in fine-tuning vs. SVM

# regions with CNN features (R-CNN)

## pros

- region proposals, SVM classifier and NMS are standard; here one just replaces the features (e.g. HOG) by CNN
- CNN features are 4k-dimensional, compared e.g. to 360k dimensions of previous state of the art
- **transfer learning**: network pre-trained on 1.2M ImageNet images, then ImageNet-specific 1000-way classification layer replaced by randomly initialized  $(c + 1)$ -way ( $c$  classes plus background) and **fine-tuning**

## cons

- **slow** (13s/image): image warped and forwarded through network for each of the  $\sim 2000$  region proposals
- **4 stages**: region extraction, CNN features, SVM classifier, regressor
- positives/negatives defined differently in fine-tuning vs. SVM

# regions with CNN features (R-CNN)

## pros

- region proposals, SVM classifier and NMS are standard; here one just replaces the features (e.g. HOG) by CNN
- CNN features are 4k-dimensional, compared e.g. to 360k dimensions of previous state of the art
- **transfer learning**: network pre-trained on 1.2M ImageNet images, then ImageNet-specific 1000-way classification layer replaced by randomly initialized  $(c + 1)$ -way ( $c$  classes plus background) and **fine-tuning**

## cons

- **slow** (13s/image): image warped and forwarded through network for each of the  $\sim 2000$  region proposals
- **4 stages**: region extraction, CNN features, SVM classifier, regressor
- positives/negatives defined differently in fine-tuning vs. SVM

# regions with CNN features (R-CNN)

## pros

- region proposals, SVM classifier and NMS are standard; here one just replaces the features (e.g. HOG) by CNN
- CNN features are 4k-dimensional, compared e.g. to 360k dimensions of previous state of the art
- **transfer learning**: network pre-trained on 1.2M ImageNet images, then ImageNet-specific 1000-way classification layer replaced by randomly initialized  $(c + 1)$ -way ( $c$  classes plus background) and **fine-tuning**

## cons

- **slow** (13s/image): image warped and forwarded through network for each of the  $\sim 2000$  region proposals
- **4 stages**: region extraction, CNN features, SVM classifier, regressor
- positives/negatives defined differently in fine-tuning vs. SVM

# bounding box regression

- at **training**, given **proposed** and **ground truth** region  $p, g \in \mathbb{R}^4$ , define **normalized** target  $t$  for region center  $(x, y)$  and size  $(w, h)$

$$\begin{aligned} t_x &= (g_x - p_x)/p_w & t_w &= \log(g_w/p_w) \\ t_y &= (g_y - p_y)/p_h & t_h &= \log(g_h/p_h) \end{aligned}$$

- for  $j \in \{x, y, w, h\}$ , learn mapping  $y_j = f_j(p)$  according to **least squares** loss

$$L(y_j, t_j) = (y_j - t_j)^2$$

- at **inference**, given proposal  $p$ , predict region  $\hat{p}$  according to

$$\begin{aligned} \hat{p}_x &= p_w f_x(p) + p_x & \hat{p}_w &= p_w \exp(f_w(p)) \\ \hat{p}_y &= p_h f_y(p) + p_y & \hat{p}_h &= p_h \exp(f_h(p)) \end{aligned}$$

# bounding box regression

- at **training**, given **proposed** and **ground truth** region  $p, g \in \mathbb{R}^4$ , define **normalized** target  $t$  for region center  $(x, y)$  and size  $(w, h)$

$$\begin{aligned} t_x &= (g_x - p_x)/p_w & t_w &= \log(g_w/p_w) \\ t_y &= (g_y - p_y)/p_h & t_h &= \log(g_h/p_h) \end{aligned}$$

- for  $j \in \{x, y, w, h\}$ , learn mapping  $y_j = f_j(p)$  according to **least squares** loss

$$L(y_j, t_j) = (y_j - t_j)^2$$

- at **inference**, given proposal  $p$ , predict region  $\hat{p}$  according to

$$\begin{aligned} \hat{p}_x &= p_w f_x(p) + p_x & \hat{p}_w &= p_w \exp(f_w(p)) \\ \hat{p}_y &= p_h f_y(p) + p_y & \hat{p}_h &= p_h \exp(f_h(p)) \end{aligned}$$

## bounding box regression

- at **training**, given **proposed** and **ground truth** region  $p, g \in \mathbb{R}^4$ , define **normalized** target  $t$  for region center  $(x, y)$  and size  $(w, h)$

$$\begin{aligned} t_x &= (g_x - p_x)/p_w & t_w &= \log(g_w/p_w) \\ t_y &= (g_y - p_y)/p_h & t_h &= \log(g_h/p_h) \end{aligned}$$

- for  $j \in \{x, y, w, h\}$ , learn mapping  $y_j = f_j(p)$  according to **least squares** loss

$$L(y_j, t_j) = (y_j - t_j)^2$$

- at **inference**, given proposal  $p$ , predict region  $\hat{p}$  according to

$$\begin{aligned} \hat{p}_x &= p_w f_x(p) + p_x & \hat{p}_w &= p_w \exp(f_w(p)) \\ \hat{p}_y &= p_h f_y(p) + p_y & \hat{p}_h &= p_h \exp(f_h(p)) \end{aligned}$$

# spatial pyramid pooling (SPP)

[He et al. 2014]



- we need to extract features and classify each region
- we can crop or warp them to fixed size, then feed to CNN for both
- or we can extract features of arbitrary size with convolutions,  
**max-pool** features to fixed size, then classify

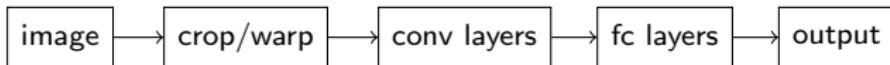
# spatial pyramid pooling (SPP)

[He et al. 2014]



crop

warp



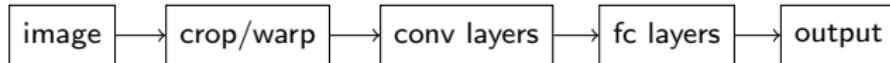
- we need to extract features and classify each region
- we can crop or warp them to fixed size, then feed to CNN for both
- or we can extract features of arbitrary size with convolutions,  
**max-pool** features to fixed size, then classify

# spatial pyramid pooling (SPP)

[He et al. 2014]



crop

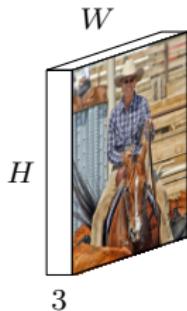


warp



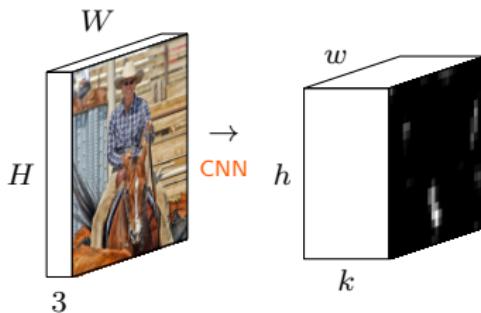
- we need to extract features and classify each region
- we can crop or warp them to fixed size, then feed to CNN for both
- or we can extract features of arbitrary size with convolutions,  
**max-pool** features to fixed size, then classify

# spatial pyramid pooling (SPP)



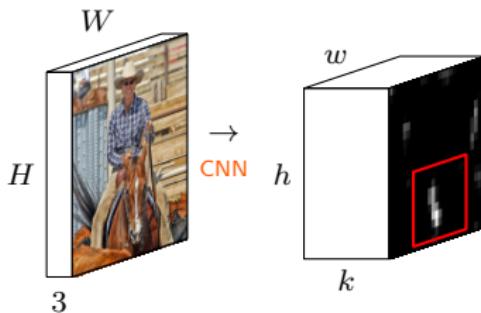
- 3-channel RGB input, arbitrary size
- input yields a single  $k$  dimensional feature map
- each region proposal projected onto feature maps
- then max-pooled into a number of fixed sizes  $1 \times 1, 2 \times 2, 4 \times 4$  etc. and concatenated into fixed-length representation
- when the pyramid has only one level, we call this RoI pooling

# spatial pyramid pooling (SPP)



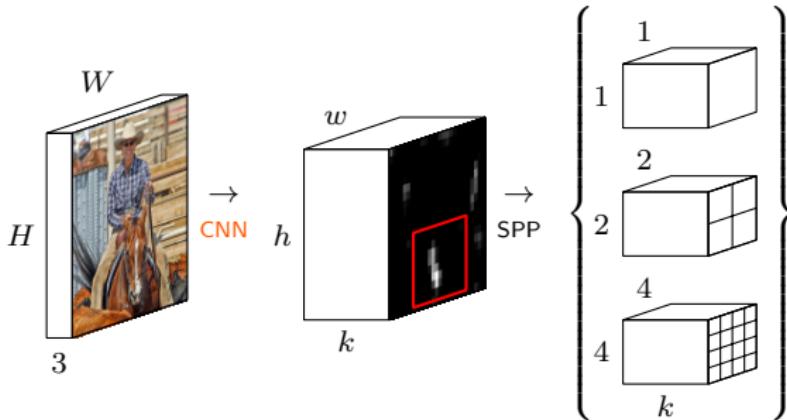
- 3-channel RGB input, arbitrary size
- input yields a single  $k$  dimensional feature map
  - each region proposal projected onto feature maps
  - then max-pooled into a number of fixed sizes  $1 \times 1, 2 \times 2, 4 \times 4$  etc. and concatenated into fixed-length representation
  - when the pyramid has only one level, we call this RoI pooling

# spatial pyramid pooling (SPP)



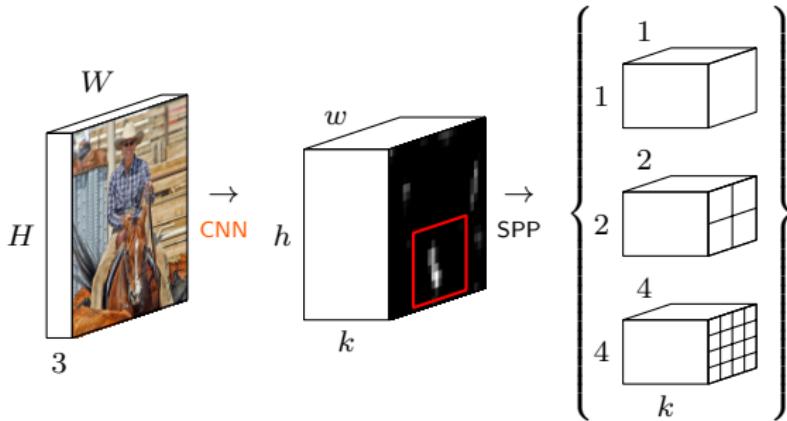
- 3-channel RGB input, **arbitrary** size
- input yields a **single**  $k$  dimensional feature map
- each region proposal projected onto feature maps
- then max-pooled into a number of **fixed sizes**  $1 \times 1, 2 \times 2, 4 \times 4$  etc.  
and concatenated into fixed-length representation
- when the pyramid has only one level, we call this **RoI pooling**

# spatial pyramid pooling (SPP)



- 3-channel RGB input, **arbitrary** size
- input yields a **single**  $k$  dimensional feature map
- each region proposal projected onto feature maps
- then max-pooled into a number of **fixed sizes**  $1 \times 1, 2 \times 2, 4 \times 4$  etc.  
and concatenated into fixed-length representation
- when the pyramid has only one level, we call this **RoI pooling**

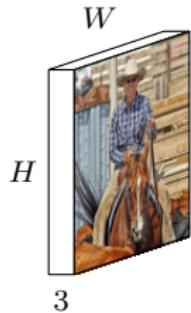
# spatial pyramid pooling (SPP)



- 3-channel RGB input, **arbitrary** size
- input yields a **single**  $k$  dimensional feature map
- each region proposal projected onto feature maps
- then max-pooled into a number of **fixed sizes**  $1 \times 1, 2 \times 2, 4 \times 4$  etc.  
and concatenated into fixed-length representation
- when the pyramid has only one level, we call this **RoI pooling**

# fast R-CNN (FRCN)

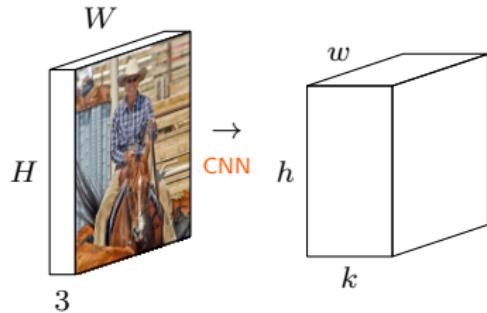
[Girshick 2015]



- 3-channel RGB input, arbitrary size
- input yields a single  $k = 4096$  dimensional feature map by VGG-16
- $\sim 2000$  region proposals, projected onto feature maps and RoI-pooled into fixed size  $w' \times h' \times k = 7 \times 7 \times k$
- several fully-connected layers follow, for each pooled map
- each pooled map is classified into  $c + 1$  classes ( $c$  + background) by single softmax and localized by bounding box regression

# fast R-CNN (FRCN)

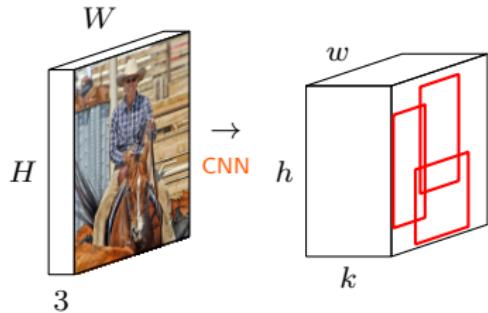
[Girshick 2015]



- 3-channel RGB input, arbitrary size
- input yields a single  $k = 4096$  dimensional feature map by VGG-16
- $\sim 2000$  region proposals, projected onto feature maps and RoI-pooled into fixed size  $w' \times h' \times k = 7 \times 7 \times k$
- several fully-connected layers follow, for each pooled map
- each pooled map is classified into  $c + 1$  classes ( $c$  + background) by single softmax and localized by bounding box regression

# fast R-CNN (FRCN)

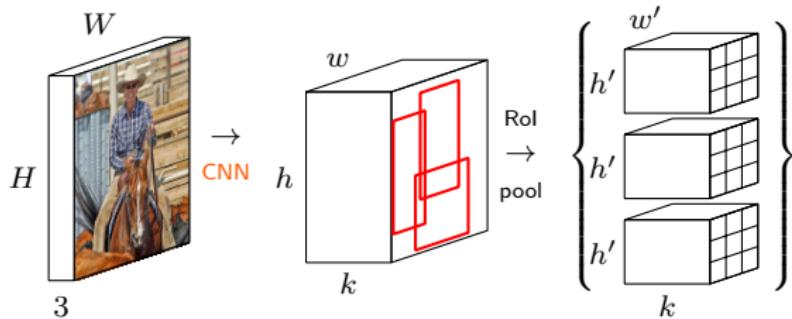
[Girshick 2015]



- 3-channel RGB input, arbitrary size
- input yields a single  $k = 4096$  dimensional feature map by VGG-16
- $\sim 2000$  region proposals, projected onto feature maps and RoI-pooled into fixed size  $w' \times h' \times k = 7 \times 7 \times k$
- several fully-connected layers follow, for each pooled map
- each pooled map is classified into  $c + 1$  classes ( $c$  + background) by single softmax and localized by bounding box regression

# fast R-CNN (FRCN)

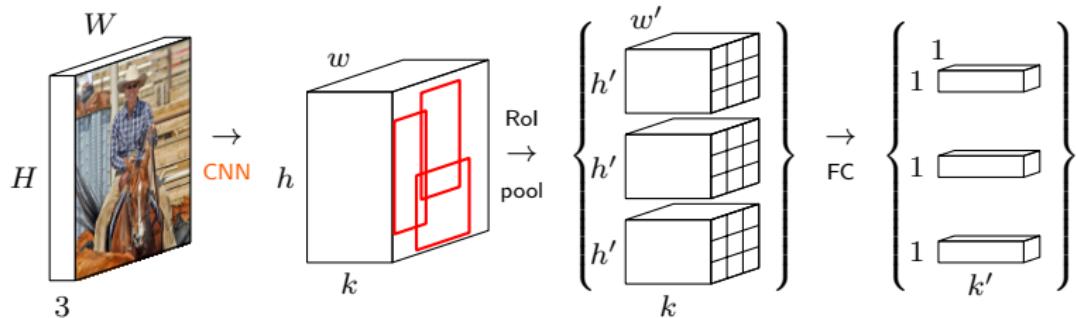
[Girshick 2015]



- 3-channel RGB input, **arbitrary** size
- input yields a **single**  $k = 4096$  dimensional feature map by VGG-16
- $\sim 2000$  region proposals, projected onto feature maps and RoI-pooled into **fixed size**  $w' \times h' \times k = 7 \times 7 \times k$
- several fully-connected layers follow, **for each** pooled map
- each pooled map is classified into  $c + 1$  classes ( $c$  + background) by **single softmax** and localized by bounding box regression

# fast R-CNN (FRCN)

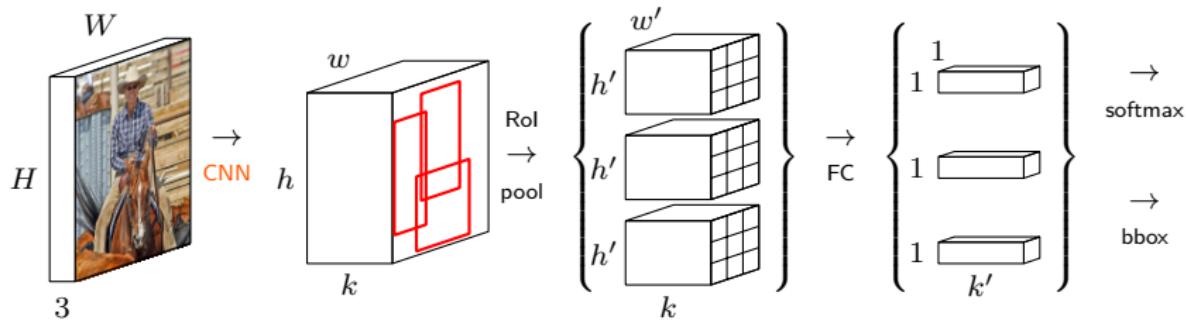
[Girshick 2015]



- 3-channel RGB input, **arbitrary** size
- input yields a **single**  $k = 4096$  dimensional feature map by VGG-16
- $\sim 2000$  region proposals, projected onto feature maps and RoI-pooled into **fixed size**  $w' \times h' \times k = 7 \times 7 \times k$
- several fully-connected layers follow, **for each** pooled map
- each pooled map is classified into  $c + 1$  classes ( $c +$  background) by **single softmax** and localized by bounding box regression

# fast R-CNN (FRCN)

[Girshick 2015]



- 3-channel RGB input, **arbitrary** size
- input yields a **single**  $k = 4096$  dimensional feature map by VGG-16
- $\sim 2000$  region proposals, projected onto feature maps and RoI-pooled into **fixed size**  $w' \times h' \times k = 7 \times 7 \times k$
- several fully-connected layers follow, **for each** pooled map
- each pooled map is classified into  $c + 1$  classes ( $c + \text{background}$ ) by **single softmax** and localized by bounding box regression

# fast R-CNN (FRCN)

## pros

- **fast** (0.32s/image;  $9\times$  training,  $213\times$  test speedup vs. R-CNN): image forwarded through network only once, only few layers are region-specific
- **2 stages**: only region proposals are separate; features, classifier and regressor are trained end-to-end with **multi-task** loss
- better performance

## cons

- region proposals are still needed for performance, but are now the **bottleneck** ( $\sim 2$ s/image)
- **single-scale**

# fast R-CNN (FRCN)

## pros

- **fast** (0.32s/image;  $9\times$  training,  $213\times$  test speedup vs. R-CNN): image forwarded through network only once, only few layers are region-specific
- **2 stages**: only region proposals are separate; features, classifier and regressor are trained end-to-end with **multi-task** loss
- better performance

## cons

- region proposals are still needed for performance, but are now the **bottleneck** ( $\sim 2$ s/image)
- **single-scale**

# fast R-CNN (FRCN)

## pros

- **fast** (0.32s/image;  $9\times$  training,  $213\times$  test speedup vs. R-CNN): image forwarded through network only once, only few layers are region-specific
- **2 stages**: only region proposals are separate; features, classifier and regressor are trained end-to-end with **multi-task** loss
- better performance

## cons

- region proposals are still needed for performance, but are now the **bottleneck** ( $\sim 2$ s/image)
- **single-scale**

# fast R-CNN (FRCN)

## pros

- **fast** (0.32s/image;  $9\times$  training,  $213\times$  test speedup vs. R-CNN): image forwarded through network only once, only few layers are region-specific
- **2 stages**: only region proposals are separate; features, classifier and regressor are trained end-to-end with **multi-task** loss
- better performance

## cons

- region proposals are still needed for performance, but are now the **bottleneck** ( $\sim 2$ s/image)
- **single-scale**

# fast R-CNN (FRCN)

## pros

- **fast** (0.32s/image;  $9\times$  training,  $213\times$  test speedup vs. R-CNN): image forwarded through network only once, only few layers are region-specific
- **2 stages**: only region proposals are separate; features, classifier and regressor are trained end-to-end with **multi-task** loss
- better performance

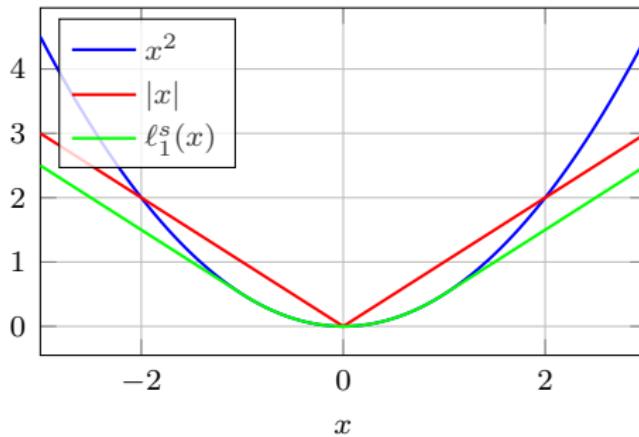
## cons

- region proposals are still needed for performance, but are now the **bottleneck** ( $\sim 2$ s/image)
- **single-scale**

# regression loss

- given region  $p$  and target  $t$ , learn mapping  $y = f(p)$  according to smooth  $\ell_1$  or Huber loss, which prevents exploding gradients

$$L(y, t) = \sum_{j \in \{x, y, h, w\}} \ell_1^s(y_j - t_j)$$
$$\ell_1^s(x) = \begin{cases} \frac{x^2}{2}, & \text{if } |x| < 1 \\ |x| - \frac{1}{2}, & \text{otherwise} \end{cases}$$



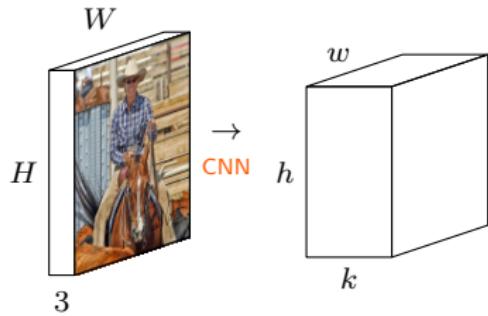
# learning object proposals: MultiBox detector\*

[Erhan et al. 2014]

- a fixed number (e.g. 100 or 200) of **class-agnostic** object proposals are **learned** by regression on image representation
- this is **faster** than e.g. selective search
- however, proposal generation is **not** convolutional, but rather based on a fully connected layer
- the next step would be to integrate object proposals and classifier, making the pipeline **end-to-end** trainable

# faster R-CNN

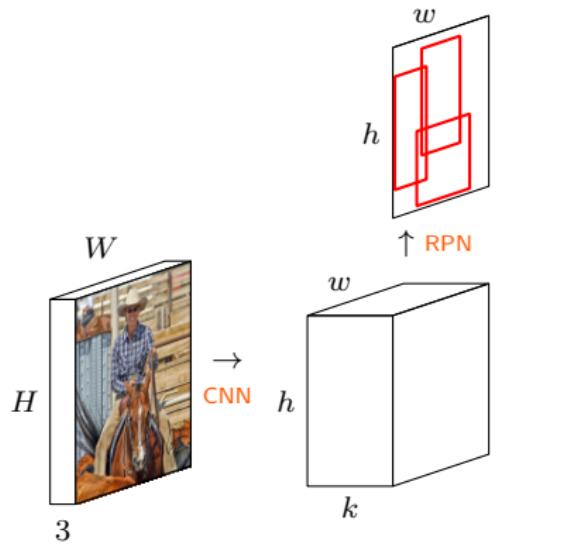
[Ren et al. 2015]



- same input, same VGG-16 feature maps as Fast R-CNN
- proposals detected directly on feature maps by RPN and max-pooled
- same classifier, same bounding box regression, but now also for RPN

# faster R-CNN

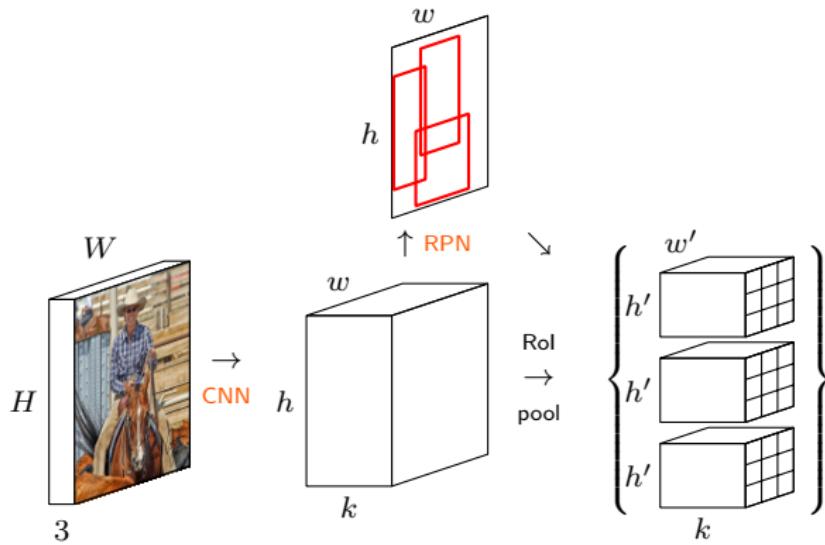
[Ren et al. 2015]



- same input, same VGG-16 feature maps as Fast R-CNN
- proposals detected **directly on feature maps** by RPN and max-pooled
- same classifier, same bounding box regression, but now also for RPN

# faster R-CNN

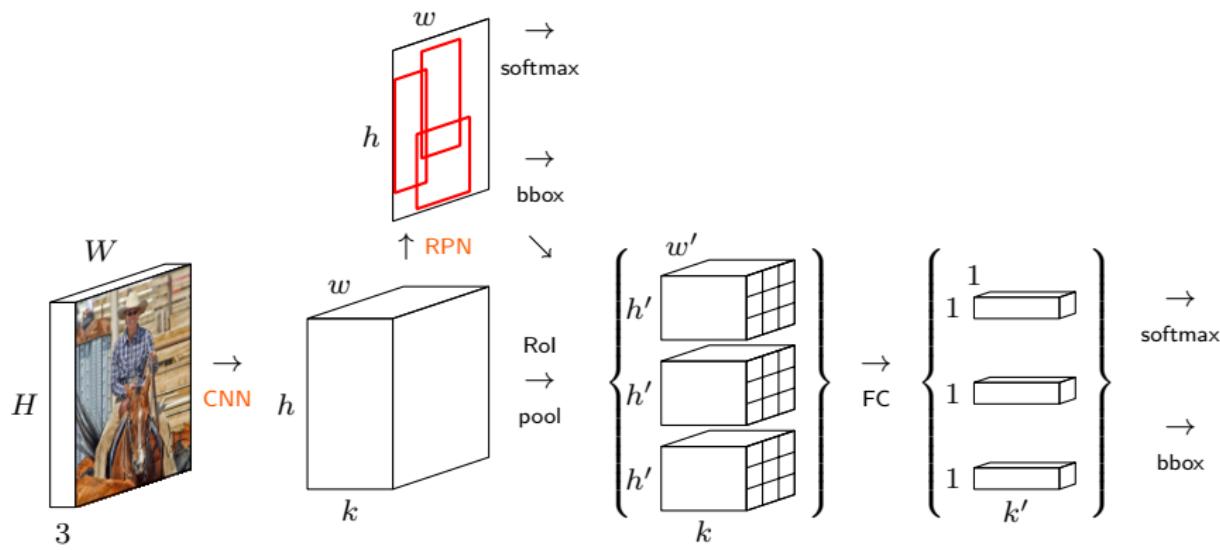
[Ren et al. 2015]



- same input, same VGG-16 feature maps as Fast R-CNN
- proposals detected **directly on feature maps** by RPN and max-pooled
- same classifier, same bounding box regression, but now also for RPN

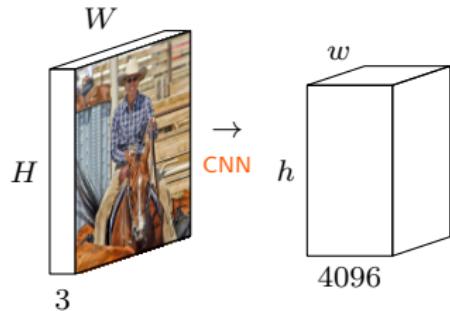
# faster R-CNN

[Ren et al. 2015]



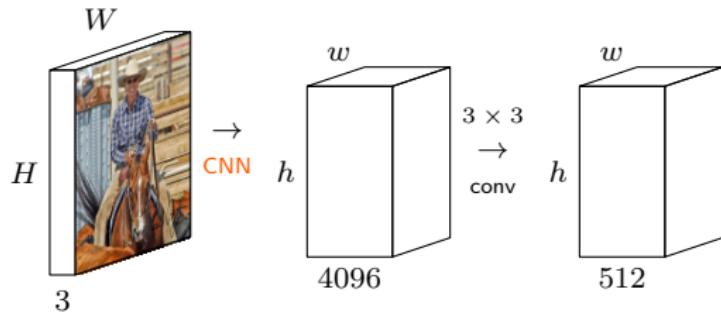
- same input, same VGG-16 feature maps as Fast R-CNN
- proposals detected **directly on feature maps** by RPN and max-pooled
- same classifier, same bounding box regression, but now also for RPN

# region proposal network (RPN)



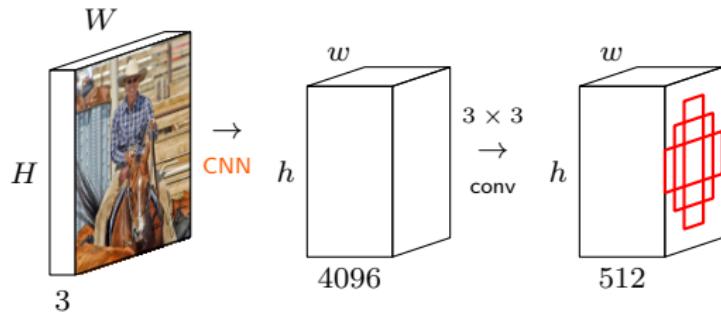
- same input, same feature maps, dimension reduced to 512
- $a = 9$  anchors at each position, for 3 scales and 3 aspect ratios
- $2a$  classification (object/non-object) scores and  $4a$  bounding box coordinates relative to anchor at each position
- softmax on scores, regression loss on coordinates
- region proposals by non-maxima suppression

# region proposal network (RPN)



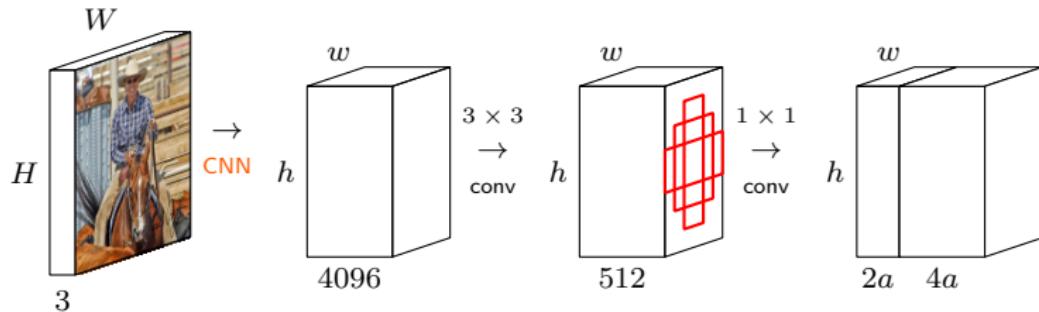
- same input, same feature maps, dimension reduced to 512
- $a = 9$  anchors at each position, for 3 scales and 3 aspect ratios
- $2a$  classification (object/non-object) scores and  $4a$  bounding box coordinates relative to anchor at each position
- softmax on scores, regression loss on coordinates
- region proposals by non-maxima suppression

# region proposal network (RPN)



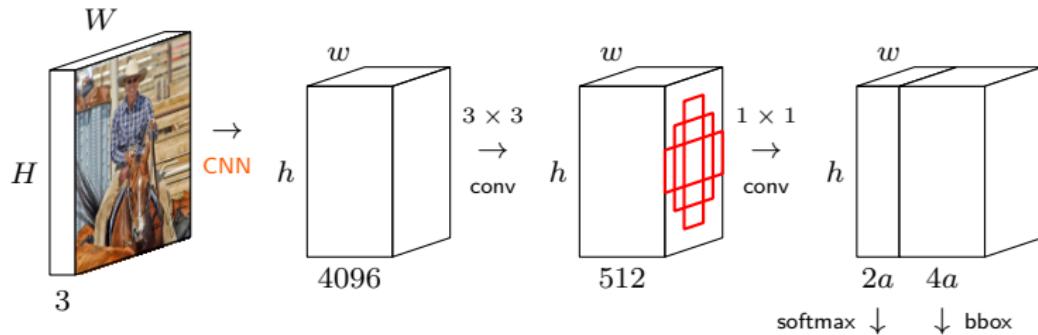
- same input, same feature maps, dimension reduced to 512
- $a = 9$  anchors at each position, for 3 scales and 3 aspect ratios
- $2a$  classification (object/non-object) scores and  $4a$  bounding box coordinates relative to anchor at each position
- softmax on scores, regression loss on coordinates
- region proposals by non-maxima suppression

# region proposal network (RPN)



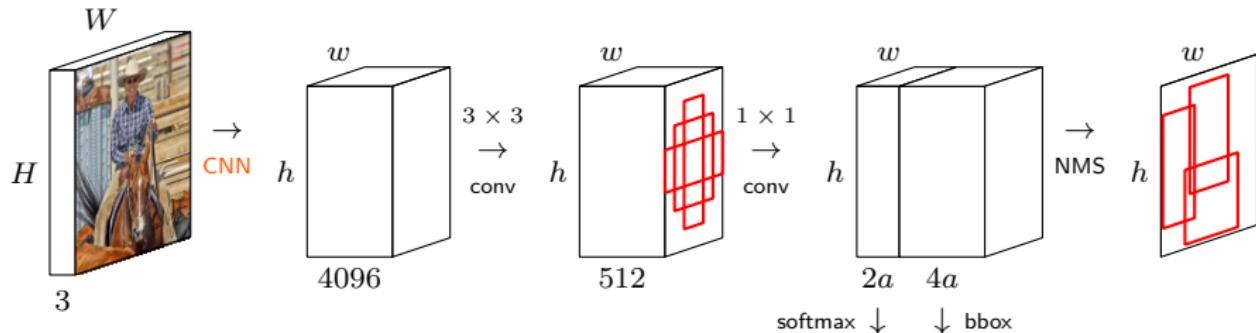
- same input, same feature maps, dimension reduced to 512
- $a = 9$  **anchors** at each position, for 3 scales and 3 aspect ratios
- $2a$  classification (object/non-object) **scores** and  $4a$  bounding box **coordinates** relative to anchor at each position
- softmax on scores, regression loss on coordinates
- region proposals by **non-maxima suppression**

# region proposal network (RPN)



- same input, same feature maps, dimension reduced to 512
- $a = 9$  anchors at each position, for 3 scales and 3 aspect ratios
- $2a$  classification (object/non-object) scores and  $4a$  bounding box coordinates relative to anchor at each position
- softmax on scores, regression loss on coordinates
- region proposals by non-maxima suppression

# region proposal network (RPN)



- same input, same feature maps, dimension reduced to 512
- $a = 9$  anchors at each position, for 3 scales and 3 aspect ratios
- $2a$  classification (object/non-object) scores and  $4a$  bounding box coordinates relative to anchor at each position
- softmax on scores, regression loss on coordinates
- region proposals by non-maxima suppression

# faster R-CNN

## pros

- faster (0.2s/image including proposals; 10× test speedup vs. fast R-CNN): only few layers are used for RPN and region-specific classification and regression
- trained end-to-end including features, region proposals, classifier and regressor
- more accurate: region proposals are learned, RPN is convolutional

## cons

- still, several fully-connected layers needed for region-specific tasks
- still single-scale

# faster R-CNN

## pros

- faster (0.2s/image including proposals; 10× test speedup vs. fast R-CNN): only few layers are used for RPN and region-specific classification and regression
- trained end-to-end including features, region proposals, classifier and regressor
- more accurate: region proposals are learned, RPN is convolutional

## cons

- still, several fully-connected layers needed for region-specific tasks
- still single-scale

# faster R-CNN

## pros

- faster (0.2s/image including proposals; 10× test speedup vs. fast R-CNN): only few layers are used for RPN and region-specific classification and regression
- trained end-to-end including features, region proposals, classifier and regressor
- more accurate: region proposals are learned, RPN is convolutional

## cons

- still, several fully-connected layers needed for region-specific tasks
- still single-scale

# faster R-CNN

## pros

- faster (0.2s/image including proposals; 10× test speedup vs. fast R-CNN): only few layers are used for RPN and region-specific classification and regression
- trained end-to-end including features, region proposals, classifier and regressor
- more accurate: region proposals are learned, RPN is convolutional

## cons

- still, several fully-connected layers needed for region-specific tasks
- still single-scale

# faster R-CNN

## pros

- faster (0.2s/image including proposals; 10× test speedup vs. fast R-CNN): only few layers are used for RPN and region-specific classification and regression
- trained end-to-end including features, region proposals, classifier and regressor
- more accurate: region proposals are learned, RPN is convolutional

## cons

- still, several fully-connected layers needed for region-specific tasks
- still single-scale

# online hard example mining (OHEM)\*

[Shrivastava et al. 2016]

- models with separate SVM classifier (R-CNN, SPP) use **Rol-centric** mini-batches, sampled from all training images
- to enable end-to-end fine-tuning of all layers, **image-centric** mini-batches are used with very few images (1-2) but thousands of candidate regions
- most regions are negative: this class **imbalance** can overwhelm the classifier
- it is standard to use a fixed positive to negative ratio (e.g. 1:1 or 1:4)
- OHEM, instead, evaluates **all** candidate regions and samples the hardest ones, without any fixed ratio

# online hard example mining (OHEM)\*

[Shrivastava et al. 2016]

- models with separate SVM classifier (R-CNN, SPP) use **Rol-centric** mini-batches, sampled from all training images
- to enable end-to-end fine-tuning of all layers, **image-centric** mini-batches are used with very few images (1-2) but thousands of candidate regions
- most regions are negative: this class **imbalance** can overwhelm the classifier
- it is standard to use a fixed positive to negative ratio (e.g. 1:1 or 1:4)
- OHEM, instead, evaluates **all** candidate regions and samples the hardest ones, without any fixed ratio

# online hard example mining (OHEM)\*

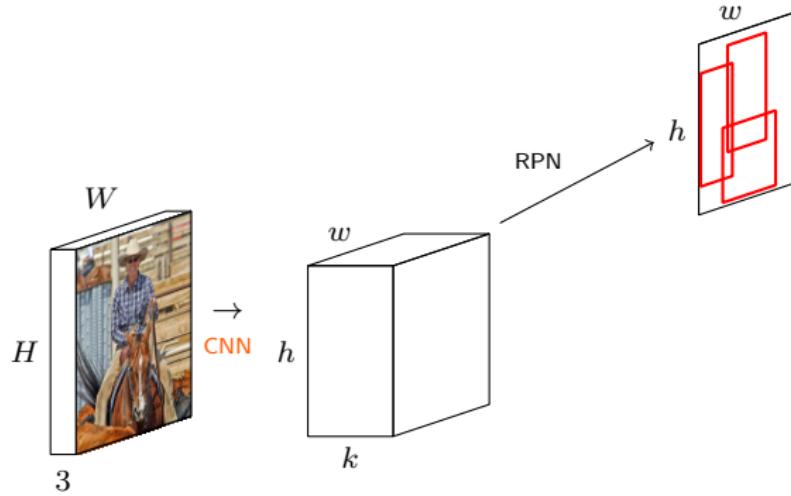
[Shrivastava et al. 2016]

- models with separate SVM classifier (R-CNN, SPP) use **Roi-centric** mini-batches, sampled from all training images
- to enable end-to-end fine-tuning of all layers, **image-centric** mini-batches are used with very few images (1-2) but thousands of candidate regions
- most regions are negative: this class **imbalance** can overwhelm the classifier
- it is standard to use a fixed positive to negative ratio (e.g. 1:1 or 1:4)
- OHEM, instead, evaluates **all** candidate regions and samples the hardest ones, without any fixed ratio

# object parts and deformation

# region-based fully convolutional network (R-FCN)

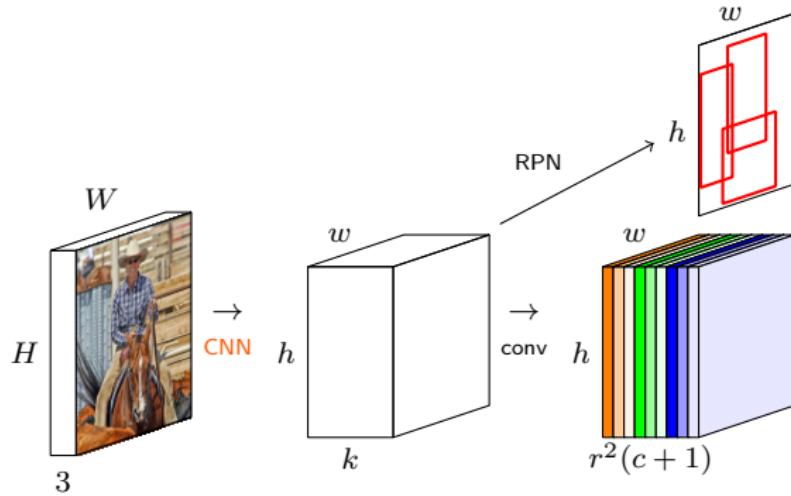
[Ren et al. 2016]



- 2048-d feature maps by ResNet-101, reduced to  $k = 1024$ , same RPN
  - $r \times r = 7 \times 7$  position-sensitive score maps per class. RoI pooling
  - similarly,  $4r^2$  position-sensitive coordinates for regression
  - no FC, just average pooling

# region-based fully convolutional network (R-FCN)

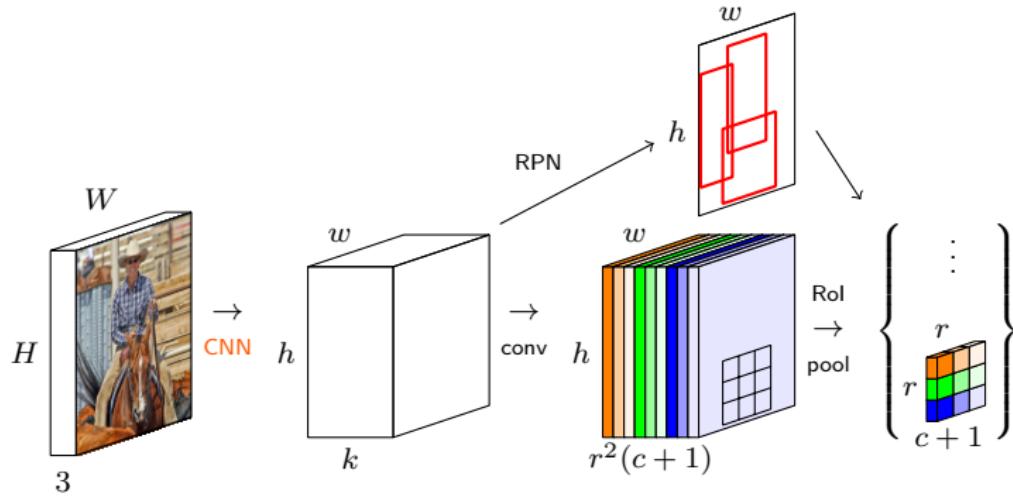
[Ren et al. 2016]



- 2048-d feature maps by ResNet-101, reduced to  $k = 1024$ , same RPN
- $r \times r = 7 \times 7$  position-sensitive score maps per class, RoI pooling
- similarly,  $4r^2$  position-sensitive coordinates for regression
- no FC, just average pooling

# region-based fully convolutional network (R-FCN)

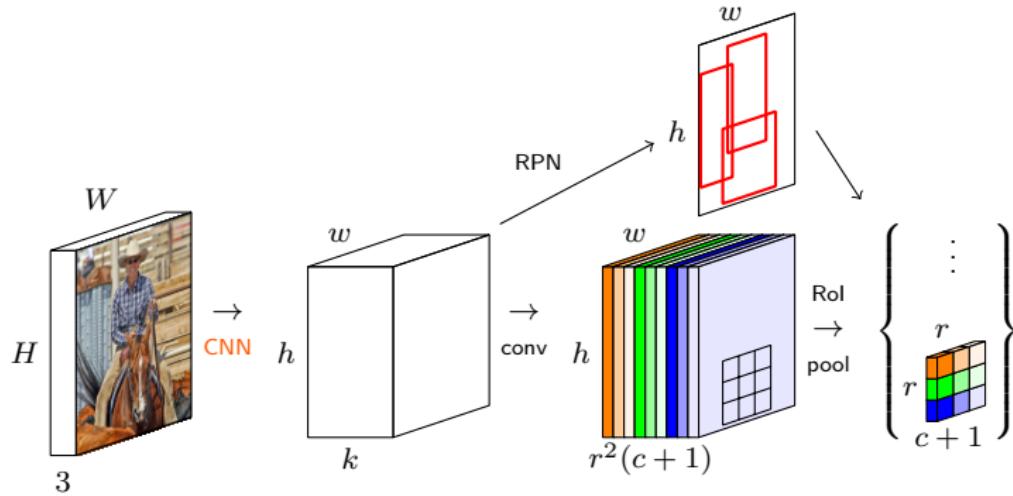
[Ren et al. 2016]



- 2048-d feature maps by ResNet-101, reduced to  $k = 1024$ , same RPN
- $r \times r = 7 \times 7$  position-sensitive score maps per class, RoI pooling
- similarly,  $4r^2$  position-sensitive coordinates for regression
- no FC, just average pooling

# region-based fully convolutional network (R-FCN)

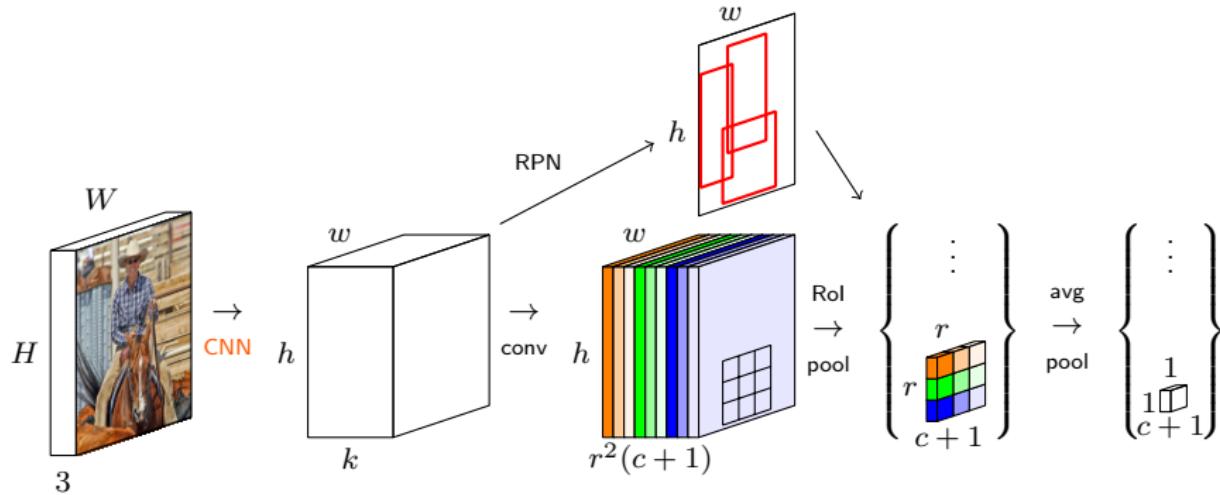
[Ren et al. 2016]



- 2048-d feature maps by ResNet-101, reduced to  $k = 1024$ , same RPN
- $r \times r = 7 \times 7$  position-sensitive score maps per class, ROI pooling
- similarly,  $4r^2$  position-sensitive coordinates for regression
- no FC, just average pooling

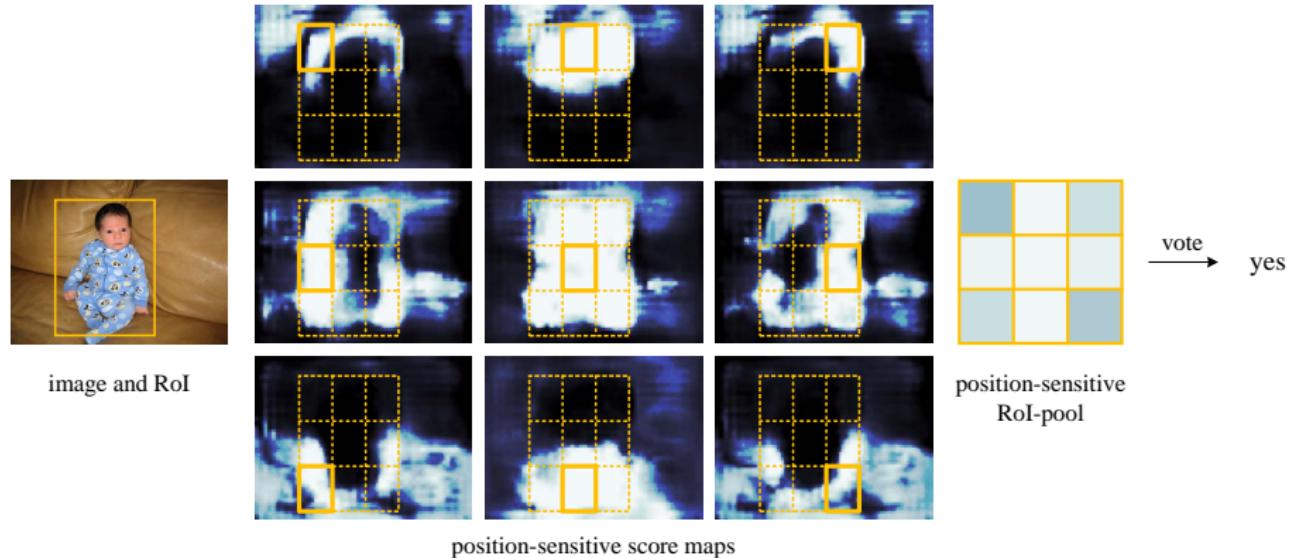
# region-based fully convolutional network (R-FCN)

[Ren et al. 2016]



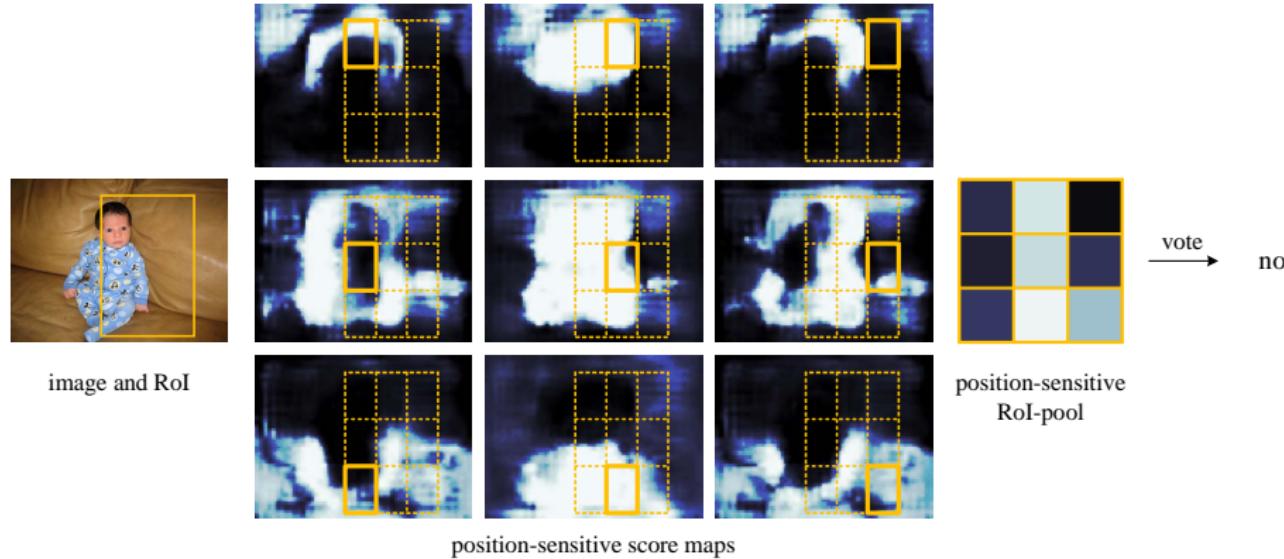
- 2048-d feature maps by ResNet-101, reduced to  $k = 1024$ , same RPN
- $r \times r = 7 \times 7$  position-sensitive score maps per class, RoI pooling
- similarly,  $4r^2$  position-sensitive coordinates for regression
- no FC, just average pooling

# position-sensitive score maps and RoI pooling



- ROI is correctly aligned with the object

# position-sensitive score maps and RoI pooling



- ROI is not correctly aligned with the object

# region-based fully convolutional network (R-FCN)

## pros

- **fully convolutional**: no more FC layers, maximum feature sharing between all tasks (RPN, classification, regression)
- still, spatial information is preserved by position-sensitive layer, improving localization accuracy
- **faster** (0.17s/image vs. 0.42 for faster R-CNN on ResNet-101)

## cons

- cells of position-sensitive RoI pooling are fixed
- still **single-scale**

# region-based fully convolutional network (R-FCN)

## pros

- **fully convolutional**: no more FC layers, maximum feature sharing between all tasks (RPN, classification, regression)
- still, spatial information is preserved by position-sensitive layer, improving localization accuracy
- faster (0.17s/image vs. 0.42 for faster R-CNN on ResNet-101)

## cons

- cells of position-sensitive RoI pooling are fixed
- still **single-scale**

# region-based fully convolutional network (R-FCN)

## pros

- **fully convolutional**: no more FC layers, maximum feature sharing between all tasks (RPN, classification, regression)
- still, spatial information is preserved by position-sensitive layer, improving localization accuracy
- **faster** (0.17s/image vs. 0.42 for faster R-CNN on ResNet-101)

## cons

- cells of position-sensitive RoI pooling are fixed
- still **single-scale**

# region-based fully convolutional network (R-FCN)

## pros

- **fully convolutional**: no more FC layers, maximum feature sharing between all tasks (RPN, classification, regression)
- still, spatial information is preserved by position-sensitive layer, improving localization accuracy
- **faster** (0.17s/image vs. 0.42 for faster R-CNN on ResNet-101)

## cons

- cells of position-sensitive RoI pooling are fixed
- still **single-scale**

# region-based fully convolutional network (R-FCN)

## pros

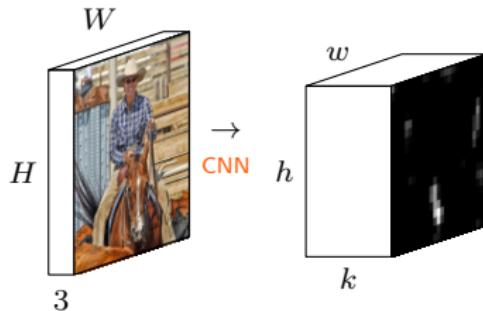
- **fully convolutional**: no more FC layers, maximum feature sharing between all tasks (RPN, classification, regression)
- still, spatial information is preserved by position-sensitive layer, improving localization accuracy
- **faster** (0.17s/image vs. 0.42 for faster R-CNN on ResNet-101)

## cons

- cells of position-sensitive RoI pooling are fixed
- still **single-scale**

# spatial transformer networks (STN)\*

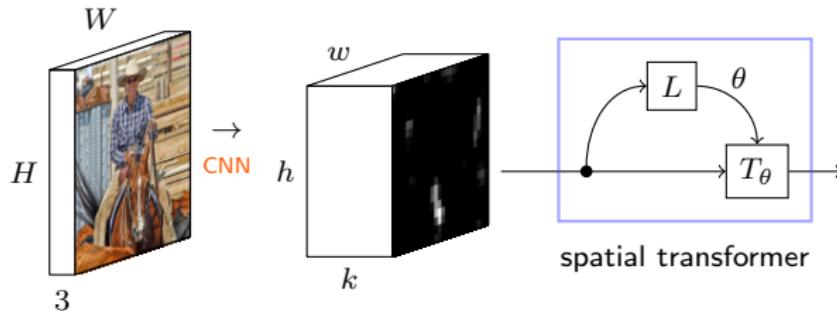
[Jaderberg et al. 2015]



- input image yields a  $k$  dimensional feature map
- a localization network  $L$  regresses a geometric transformation  $\theta$
- a transformer  $T_\theta$  applies the transformation to the feature map
- the transformation can involve resampling, cropping, even deformation
- the localization network receives no supervision other than what is backpropagated from the end task

# spatial transformer networks (STN)\*

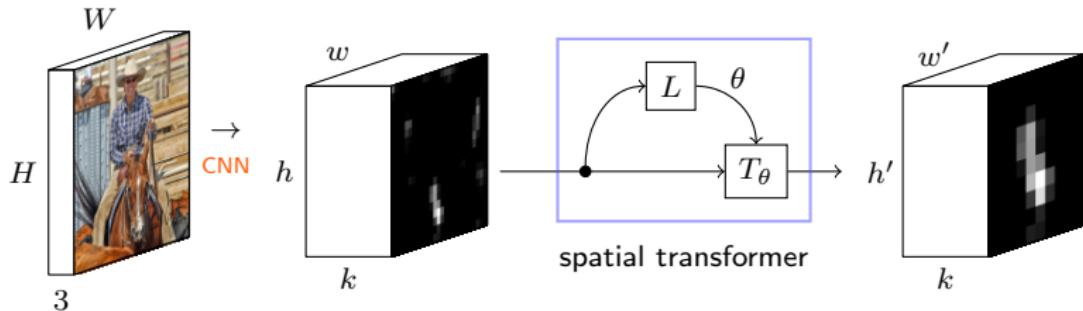
[Jaderberg et al. 2015]



- input image yields a  $k$  dimensional feature map
- a **localization network**  $L$  regresses a geometric transformation  $\theta$
- a transformer  $T_\theta$  applies the transformation to the feature map
- the transformation can involve resampling, cropping, even **deformation**
- the localization network receives **no supervision** other than what is backpropagated from the end task

# spatial transformer networks (STN)\*

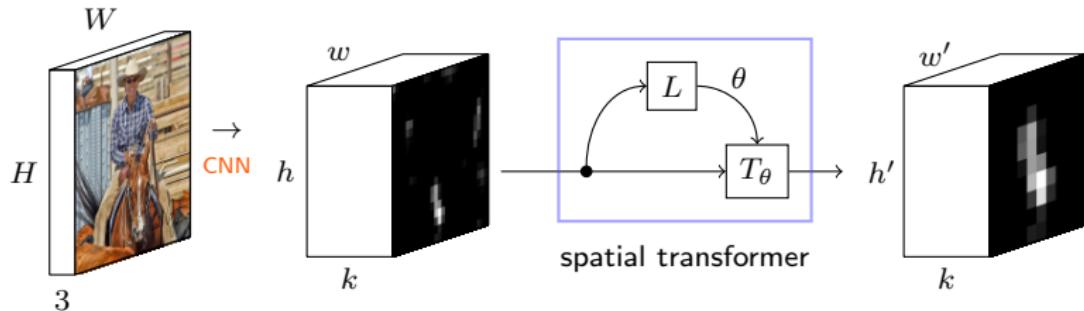
[Jaderberg et al. 2015]



- input image yields a  $k$  dimensional feature map
- a **localization network**  $L$  regresses a geometric transformation  $\theta$
- a transformer  $T_\theta$  applies the transformation to the feature map
- the transformation can involve resampling, cropping, even **deformation**
- the localization network receives **no supervision** other than what is backpropagated from the end task

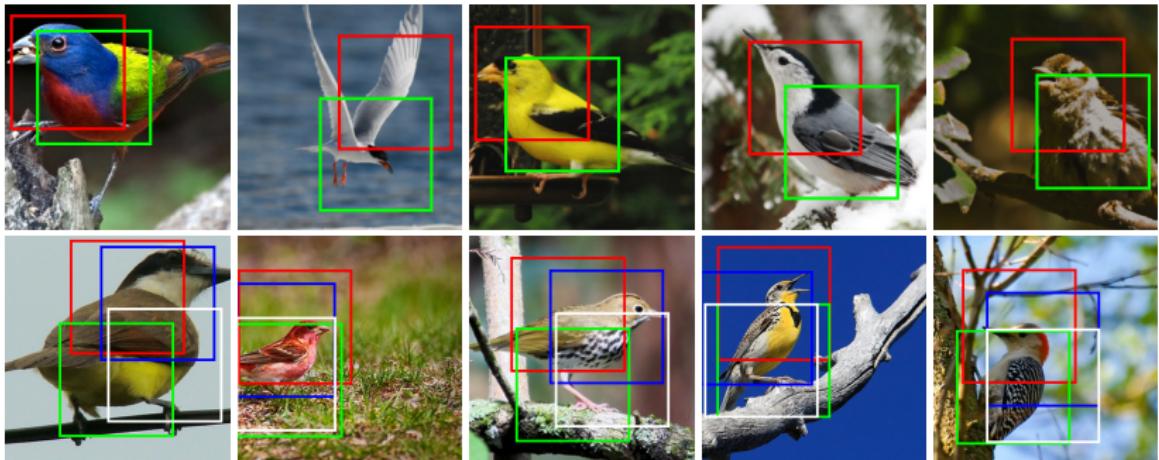
# spatial transformer networks (STN)\*

[Jaderberg et al. 2015]



- input image yields a  $k$  dimensional feature map
- a **localization network**  $L$  regresses a geometric transformation  $\theta$
- a transformer  $T_\theta$  applies the transformation to the feature map
- the transformation can involve resampling, cropping, even **deformation**
- the localization network receives **no supervision** other than what is backpropagated from the end task

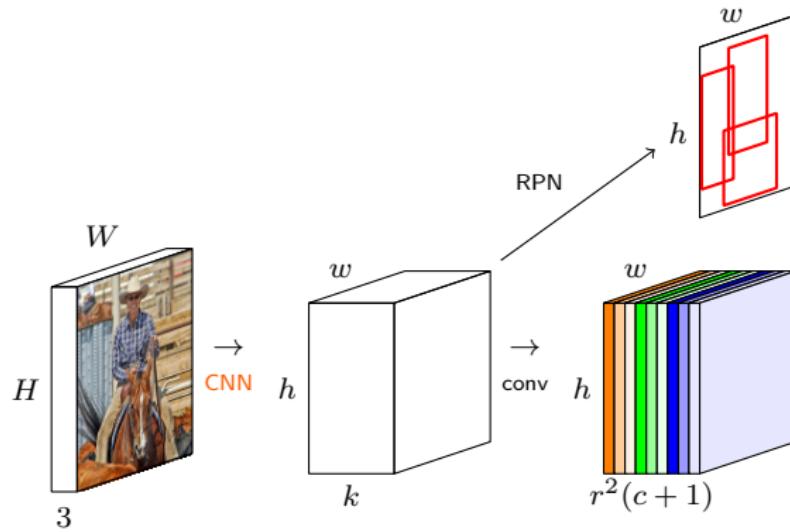
# spatial transformer networks: part learning\*



- 2 or 4 spatial transformers predict discriminative object parts with **no supervision** other than the class label
- the localization network is based on GoogLeNet and is **shared** across transformers; features are extracted by one GoogLeNet for each region
- features are concatenated and the image is classified by a single fully connected layer with softmax

# deformable RoI pooling

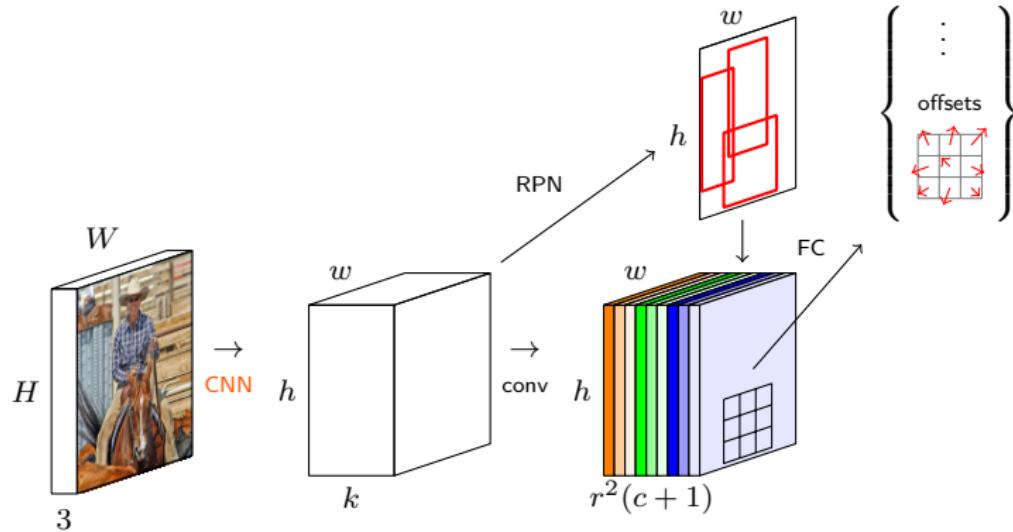
[Ren et al. 2017]



- same features, same RPN, same position-sensitive scores as R-FCN
- cell offsets by FC on RoI-pooled features, *deformable RoI pooling*
- same average pooling

# deformable RoI pooling

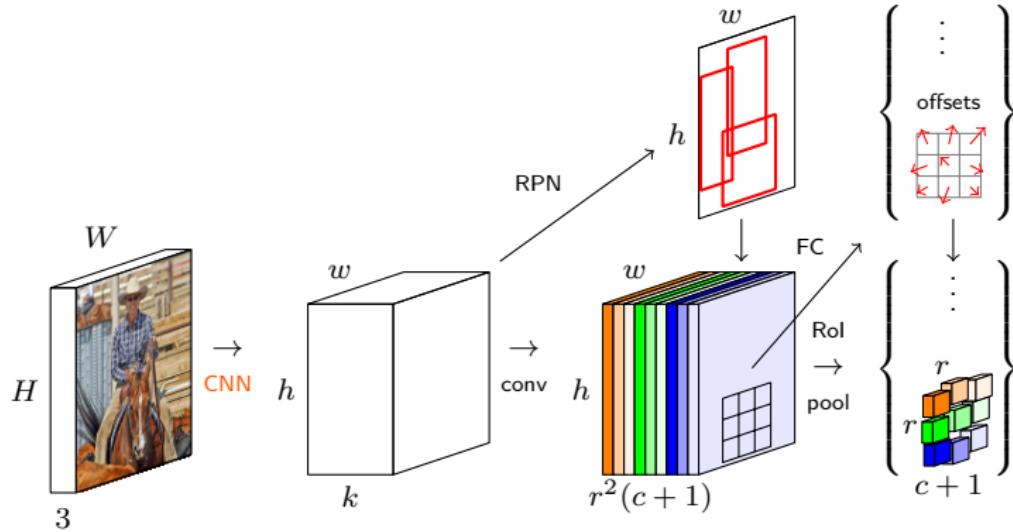
[Ren et al. 2017]



- same features, same RPN, same position-sensitive scores as R-FCN
- cell offsets by FC on RoI-pooled features, **deformable RoI pooling**
- same average pooling

# deformable RoI pooling

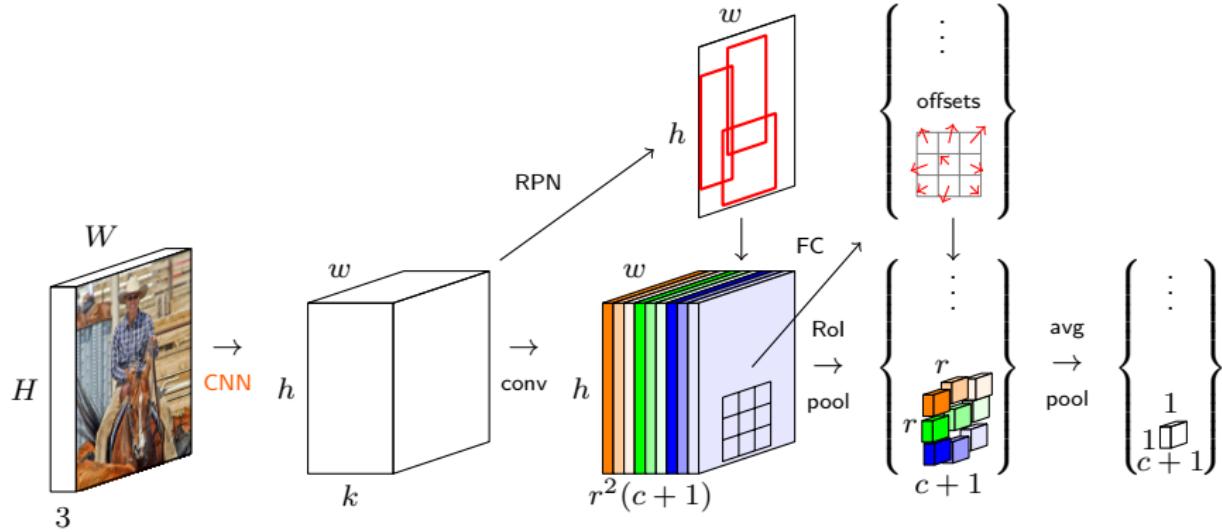
[Ren et al. 2017]



- same features, same RPN, same position-sensitive scores as R-FCN
- cell offsets by FC on RoI-pooled features, **deformable RoI pooling**
- same average pooling

# deformable RoI pooling

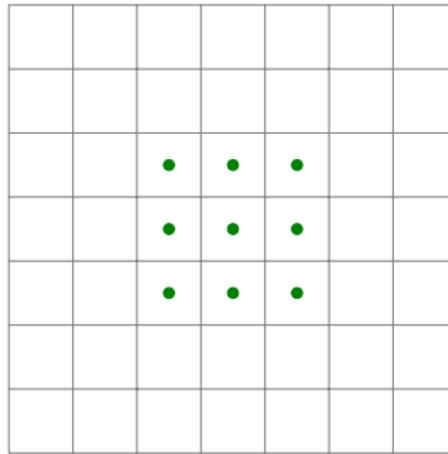
[Ren et al. 2017]



- same features, same RPN, same position-sensitive scores as R-FCN
- cell offsets by FC on RoI-pooled features, **deformable RoI pooling**
- same average pooling

# deformable convolution

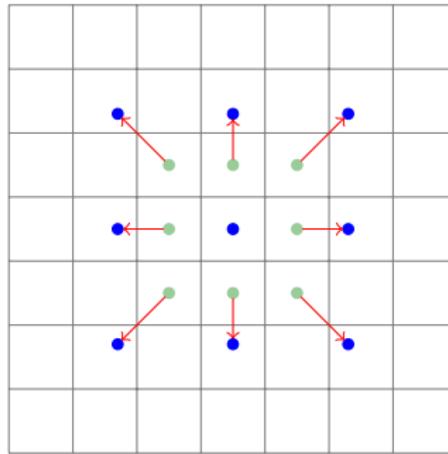
[Ren et al. 2017]



- standard convolution on  $3 \times 3$  regular sampling grid

# deformable convolution

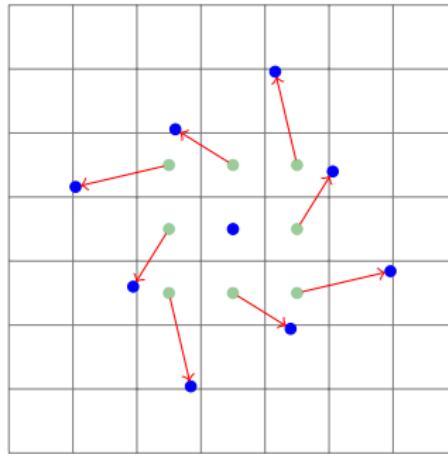
[Ren et al. 2017]



- scaled grid (as in **automatic scale selection**, but dense)

# deformable convolution

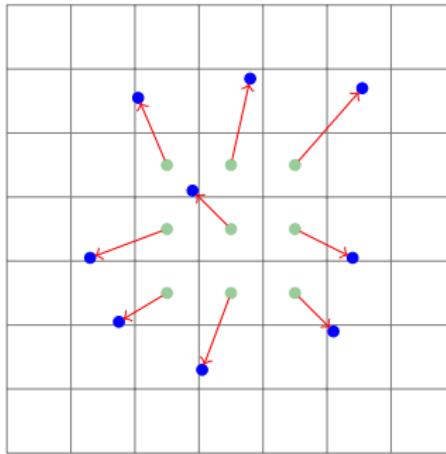
[Ren et al. 2017]



- rotated grid (as in dominant orientation selection, but dense)

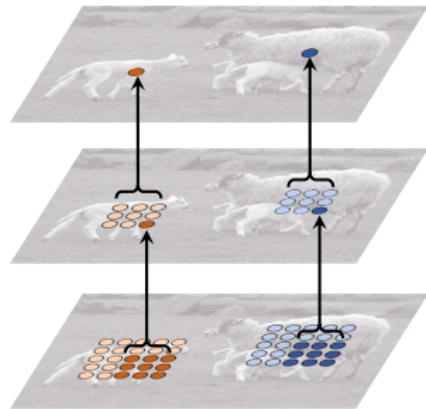
# deformable convolution

[Ren et al. 2017]



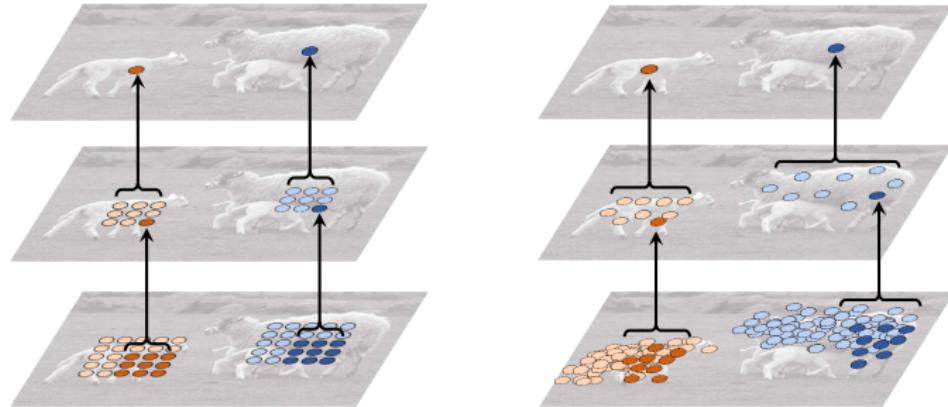
- deformed sampling grid where offsets are computed per pixel

# deformable convolution: receptive field (2 layers)



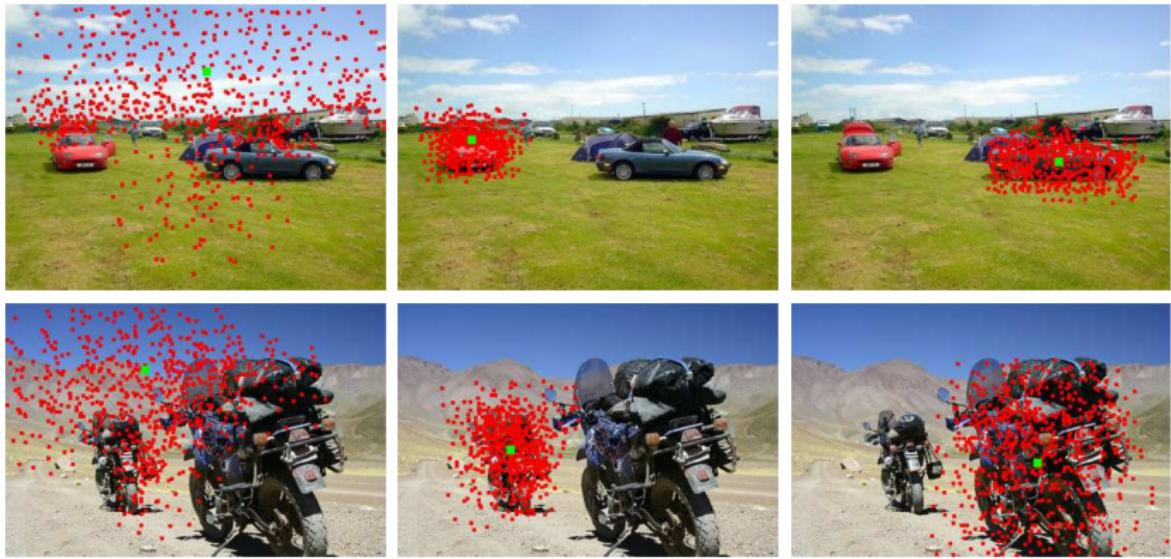
- **standard convolution**: receptive field grows with depth but only linearly, remains rectangular and is translation invariant
- **deformable convolution**: receptive field grows arbitrarily with depth, adapts per location and takes arbitrary shape

# deformable convolution: receptive field (2 layers)



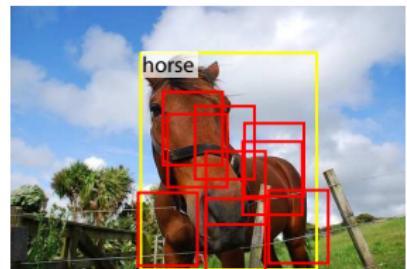
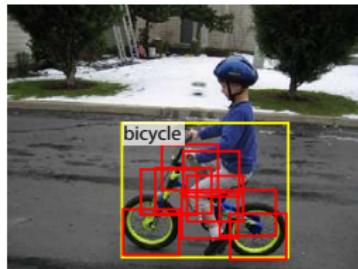
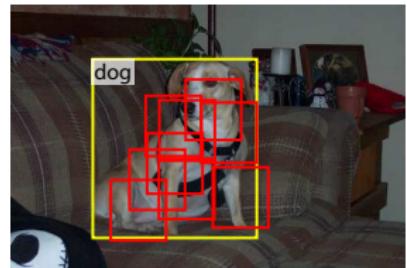
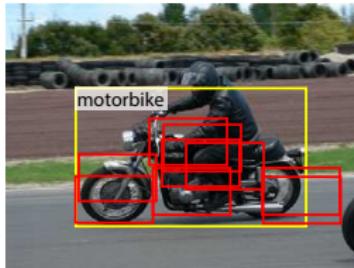
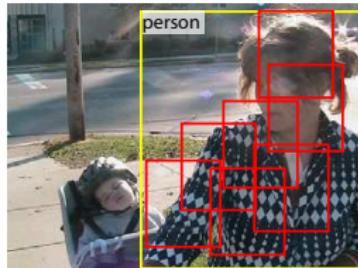
- **standard convolution:** receptive field grows with depth but only linearly, remains rectangular and is translation invariant
- **deformable convolution:** receptive field grows arbitrarily with depth, adapts per location and takes arbitrary shape

# deformable convolution: receptive field (2 layers)



- red:  $9^3 = 729$  sampling locations in 3 levels of  $3 \times 3$  deformable filters for three units (green)
- receptive field adapts to object size and shape

# deformable RoI pooling

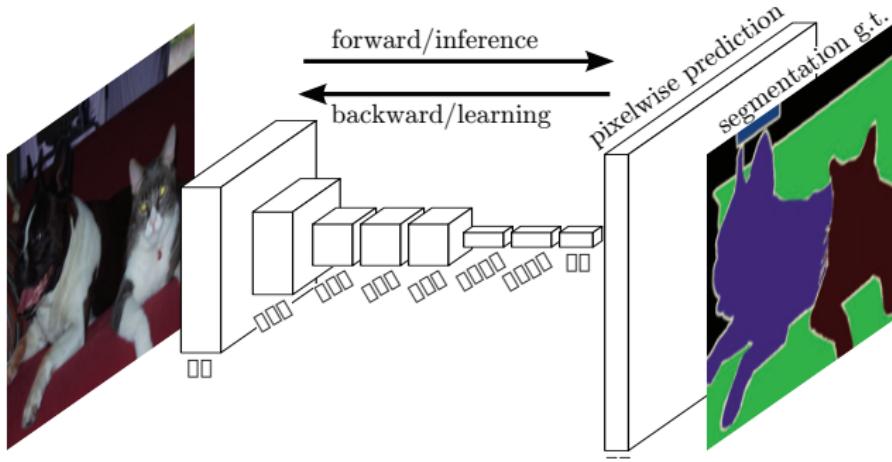


- deformed  $3 \times 3$  cells (**red**) for an input RoI (**yellow**)
- cells adapt to part locations of non-rigid objects

# scale and feature pyramids\*

# fully convolutional networks (FCN)\*

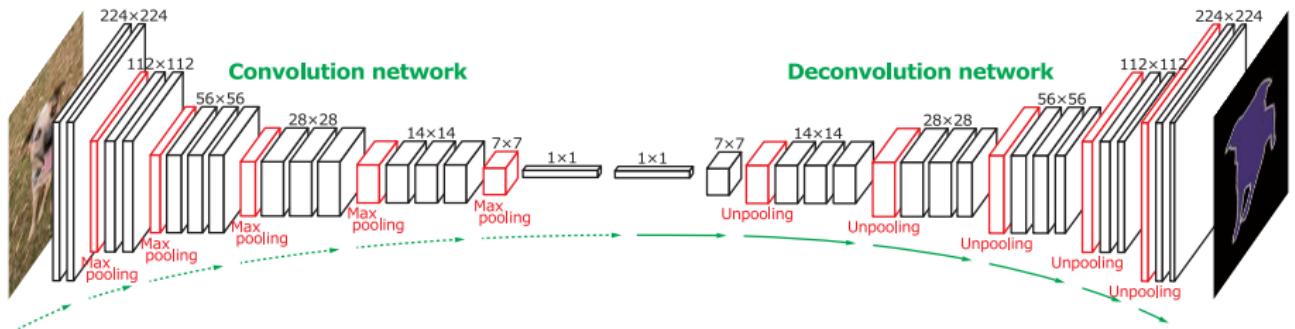
[Long et al. 2015]



- feature maps capture high-level semantics but are of low resolution
- here, they are **upsampled** to original pixel resolution
- given **pixel-wise** class label supervision, the network learns pixel-wise prediction for semantic segmentation
- there are no fully-connected layers, hence "**fully convolutional**"

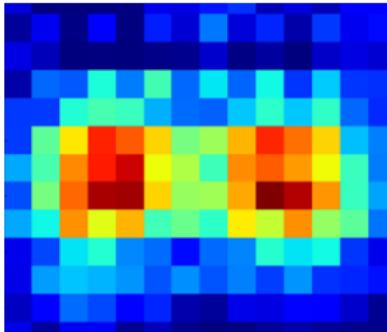
# learning to upsample\*

[Noh et al. 2015]

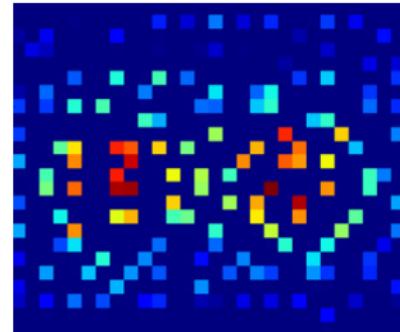


- the upsampling process is improved by **learning to invert** the max-pooling and convolution operations with **unpooling** and **deconvolution**
- **instance-wise segmentations** are obtained by applying the network to individual object proposals, as in detection

# learning to upsample



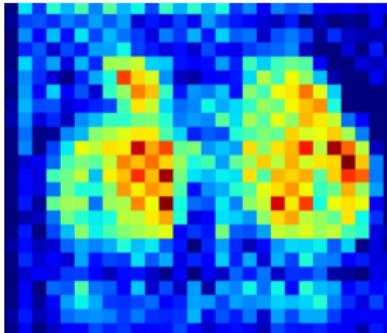
$14 \times 14$  deconv



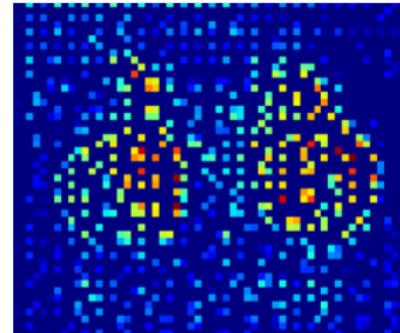
$28 \times 28$  unpool

- resolution is **decreased** from  $224 \times 224$  down to  $7 \times 7$  by five  $2 \times 2$  pooling layers and finally to  $1 \times 1$  by fully connected layer
- it is then **increased** back to  $7 \times 7$ ,  $14 \times 14$  and finally up to  $224 \times 224$  by five unpooling and deconvolution layers)
- the most appropriate feature map is chosen in each layer for visualization

## learning to upsample



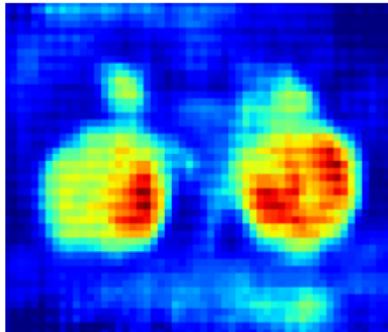
$28 \times 28$  deconv



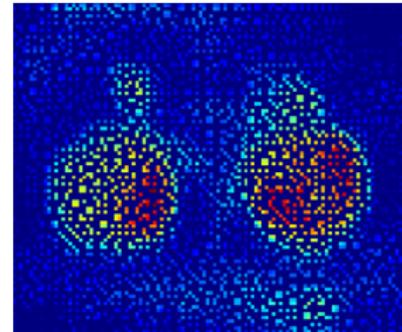
$56 \times 56$  unpool

- resolution is **decreased** from  $224 \times 224$  down to  $7 \times 7$  by five  $2 \times 2$  pooling layers and finally to  $1 \times 1$  by fully connected layer
- it is then **increased** back to  $7 \times 7$ ,  $14 \times 14$  and finally up to  $224 \times 224$  by five unpooling and deconvolution layers)
- the most appropriate feature map is chosen in each layer for visualization

## learning to upsample



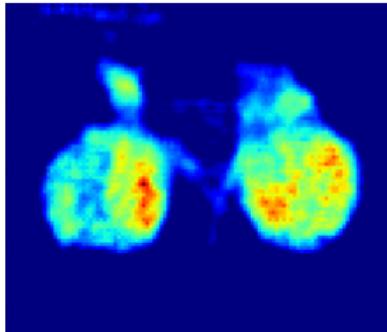
$56 \times 56$  deconv



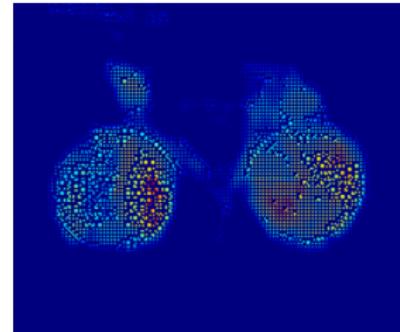
$112 \times 112$  unpool

- resolution is **decreased** from  $224 \times 224$  down to  $7 \times 7$  by five  $2 \times 2$  pooling layers and finally to  $1 \times 1$  by fully connected layer
- it is then **increased** back to  $7 \times 7$ ,  $14 \times 14$  and finally up to  $224 \times 224$  by five unpooling and deconvolution layers)
- the most appropriate feature map is chosen in each layer for visualization

## learning to upsample



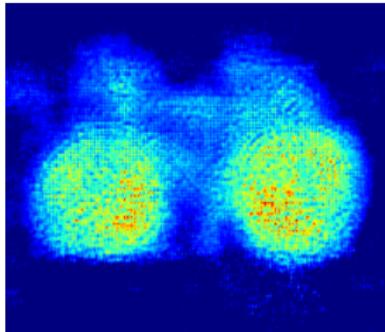
$112 \times 112$  deconv



$224 \times 224$  unpool

- resolution is **decreased** from  $224 \times 224$  down to  $7 \times 7$  by five  $2 \times 2$  pooling layers and finally to  $1 \times 1$  by fully connected layer
- it is then **increased** back to  $7 \times 7$ ,  $14 \times 14$  and finally up to  $224 \times 224$  by five unpooling and deconvolution layers)
- the most appropriate feature map is chosen in each layer for visualization

# learning to upsample



$224 \times 224$  deconv

- resolution is **decreased** from  $224 \times 224$  down to  $7 \times 7$  by five  $2 \times 2$  pooling layers and finally to  $1 \times 1$  by fully connected layer
- it is then **increased** back to  $7 \times 7$ ,  $14 \times 14$  and finally up to  $224 \times 224$  by five unpooling and deconvolution layers)
- the most appropriate feature map is chosen in each layer for visualization

## upsampling for detection

- we may not need pixel-wise prediction for detection, but we still higher resolution than e.g.  $14 \times 14$  or  $7 \times 7$  to detect and localize **small objects** accurately
- in fact, as we upsample, we will combine detections from **multiple layers** corresponding to **multiple scales**

# network “stages” or “blocks”

VGG-16

volume	
input(224, 3)	$224 \times 224 \times 3$
2× conv(3, 64, p1)	$224 \times 224 \times 64$
pool(2)	$112 \times 112 \times 64$
2× conv(3, 128, p1)	$112 \times 112 \times 128$
pool(2)	$56 \times 56 \times 128$
3× conv(3, 256, p1)	$56 \times 56 \times 256$
pool(2)	$28 \times 28 \times 256$
3× conv(3, 512, p1)	$28 \times 28 \times 512$
pool(2)	$14 \times 14 \times 512$
3× conv(3, 512, p1)	$14 \times 14 \times 512$
pool(2)	$7 \times 7 \times 512$
2× fc(4096)	4,096
fc(1000)	1,000
softmax	1,000

ResNet-101

volume	
input(224, 3)	$224 \times 224 \times 3$
conv(7, 64, p3, s2)	$112 \times 112 \times 64$
pool(3, 2, p1)	$56 \times 56 \times 64$
3× res(3, (64, 256))	$56 \times 56 \times 256$
res(3, (128, 512), s2)	$28 \times 28 \times 512$
3× res(3, (128, 512))	$28 \times 28 \times 512$
res(3, (256, 1024), s2)	$14 \times 14 \times 1024$
22× res(3, (256, 1024))	$14 \times 14 \times 1024$
res(3, (512, 2048), s2)	$7 \times 7 \times 2048$
2× res(3, (512, 2048))	$7 \times 7 \times 2048$
avg(7)	2048
fc(1000)	1000
softmax	1000

# network “stages” or “blocks”

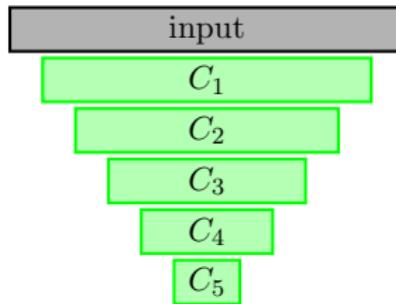
VGG-16

	volume	
	input(224, 3)	$224 \times 224 \times 3$
$C_1$	2× conv(3, 64, p1)	$224 \times 224 \times 64$
	pool(2)	$112 \times 112 \times 64$
$C_2$	2× conv(3, 128, p1)	$112 \times 112 \times 128$
	pool(2)	$56 \times 56 \times 128$
$C_3$	3× conv(3, 256, p1)	$56 \times 56 \times 256$
	pool(2)	$28 \times 28 \times 256$
$C_4$	3× conv(3, 512, p1)	$28 \times 28 \times 512$
	pool(2)	$14 \times 14 \times 512$
$C_5$	3× conv(3, 512, p1)	$14 \times 14 \times 512$
	pool(2)	$7 \times 7 \times 512$
2×	fc(4096)	4,096
	fc(1000)	1,000
	softmax	1,000

ResNet-101

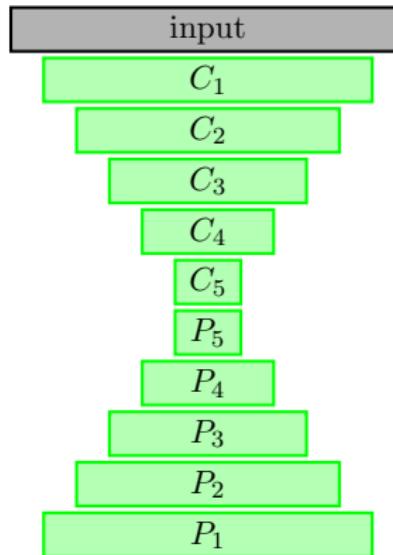
	volume	
	input(224, 3)	$224 \times 224 \times 3$
	conv(7, 64, p3, s2)	$112 \times 112 \times 64$
	pool(3, 2, p1)	$56 \times 56 \times 64$
	3× res(3, (64, 256))	$56 \times 56 \times 256$
	res(3, (128, 512), s2)	$28 \times 28 \times 512$
	3× res(3, (128, 512))	$28 \times 28 \times 512$
	res(3, (256, 1024), s2)	$14 \times 14 \times 1024$
	22× res(3, (256, 1024))	$14 \times 14 \times 1024$
	res(3, (512, 2048), s2)	$7 \times 7 \times 2048$
	2× res(3, (512, 2048))	$7 \times 7 \times 2048$
	avg(7)	2048
	fc(1000)	1000
	softmax	1000

# pyramid networks



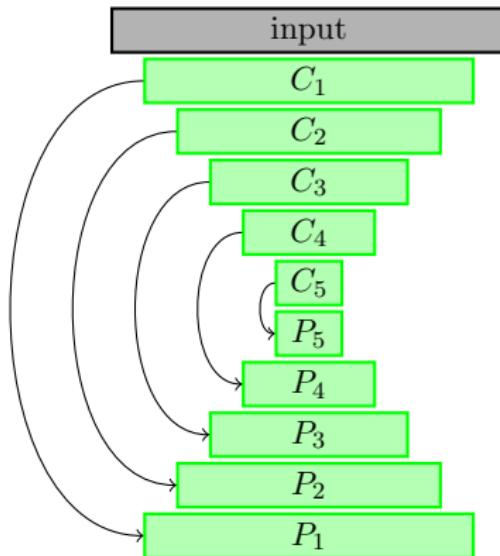
- **bottom-up** path: higher-level features, downsampling
- **top-down** path: still high-level, upsampling
- lateral connections
- predictions from multiple scales

# pyramid networks



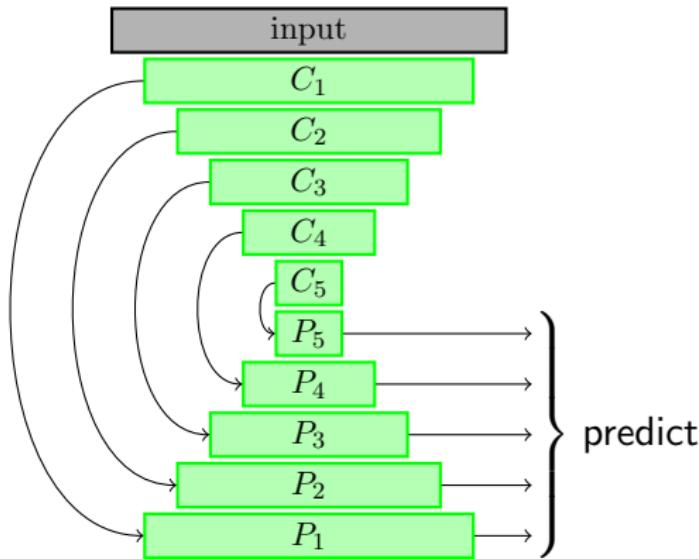
- **bottom-up** path: higher-level features, downsampling
- **top-down** path: still high-level, upsampling
- lateral connections
- predictions from multiple scales

# pyramid networks



- **bottom-up** path: higher-level features, downsampling
- **top-down** path: still high-level, upsampling
- **lateral connections**
- predictions from multiple scales

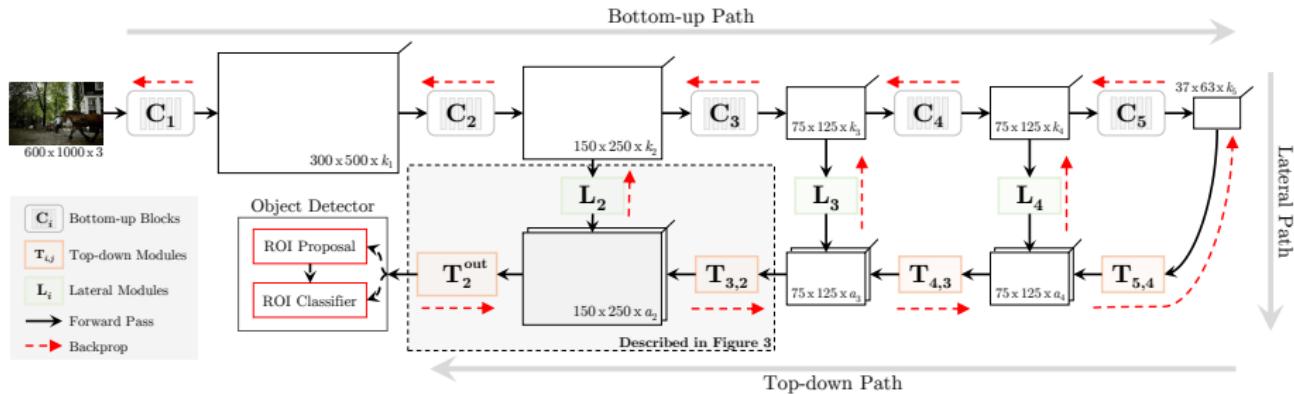
# pyramid networks



- **bottom-up** path: higher-level features, downsampling
- **top-down** path: still high-level, upsampling
- **lateral connections**
- predictions from multiple scales

# top-down modulation (TDM)\*

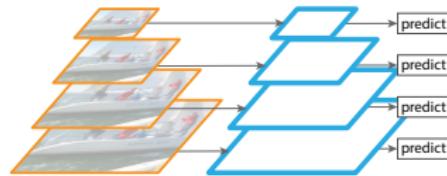
[Shrivastava et al. 2016]



- the top-down network handles the **integration** of features and attempts to influence lower-level features
- detection (or any final task) now depends on high-resolution, high-level features
- applied to VGG-16 and ResNet-101 with faster R-CNN
- however, only the final top-down module collects features

# feature pyramid networks (FPN)

[Lin et al. 2017]

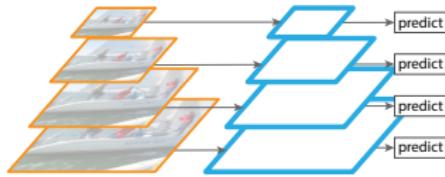


featurized image pyramid

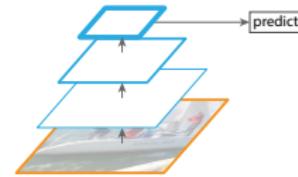
- features computed at each scale independently: **slow**
- single scale for faster detection
- reuse pyramidal feature hierarchy as if computed at different scales
- still fast, but more accurate

# feature pyramid networks (FPN)

[Lin et al. 2017]



featurized image pyramid

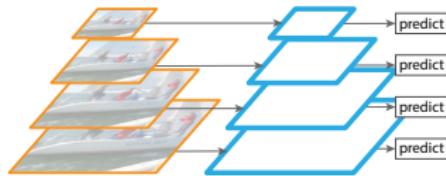


single feature map

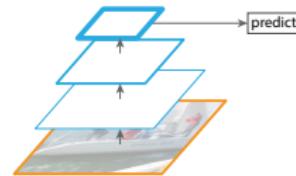
- features computed at each scale independently: **slow**
- single scale for faster detection
- reuse pyramidal feature hierarchy as if computed at different scales
- still fast, but more accurate

# feature pyramid networks (FPN)

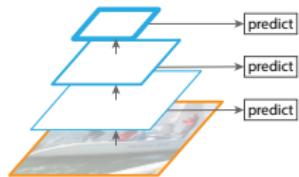
[Lin et al. 2017]



featurized image pyramid



single feature map

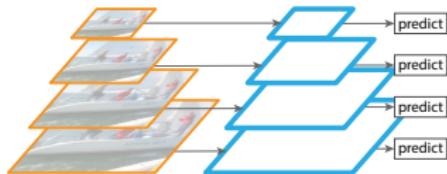


pyramidal feature hierarchy

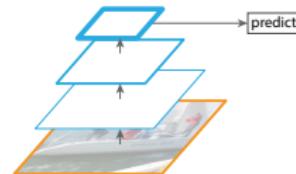
- features computed at each scale independently: **slow**
- single scale for faster detection
- reuse pyramidal feature hierarchy as if computed at different scales
- still fast, but more accurate

# feature pyramid networks (FPN)

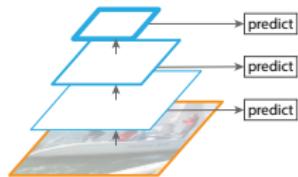
[Lin et al. 2017]



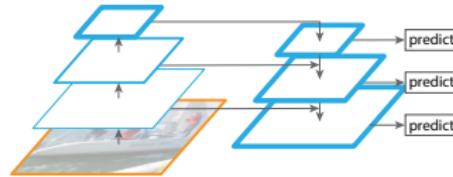
featurized image pyramid



single feature map



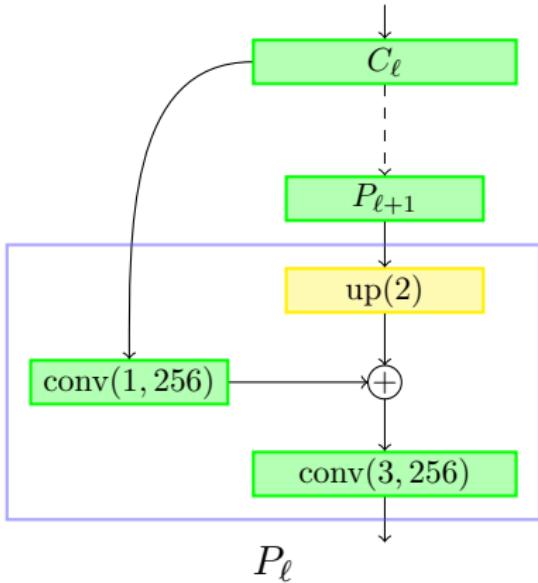
pyramidal feature hierarchy



feature pyramid network

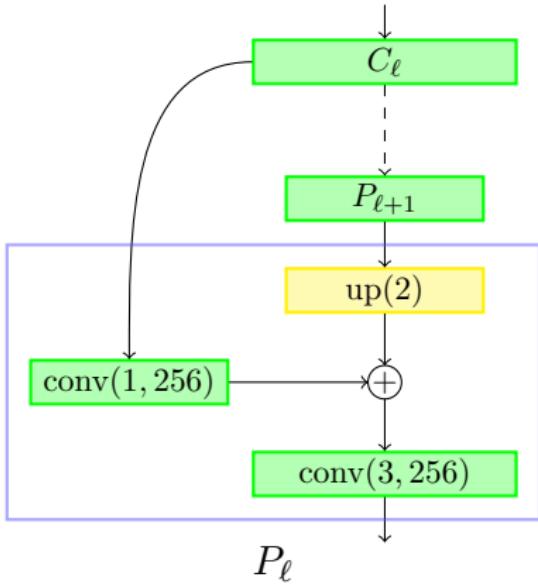
- features computed at each scale independently: **slow**
- single scale for faster detection
- reuse pyramidal feature hierarchy as if computed at different scales
- still fast, but more accurate

# feature pyramid networks (FPN)



- all top-down layers have 256 features
- top-down network initialized at  $P_5$  by  $1 \times 1$  convolution on  $C_5$
- $1 \times 1$  convolution on lateral connection reduces width
- $3 \times 3$  convolution on merged path reduces aliasing
- applied to ResNet-101 with fast/faster R-CNN
- regions are detected at all levels of top-down pyramid
- classifiers/regressors are shared across all levels

# feature pyramid networks (FPN)



- all top-down layers have 256 features
- top-down network initialized at  $P_5$  by  $1 \times 1$  convolution on  $C_5$
- $1 \times 1$  convolution on lateral connection reduces width
- $3 \times 3$  convolution on merged path reduces aliasing
- applied to ResNet-101 with **fast/faster R-CNN**
- regions are detected at **all levels** of top-down pyramid
- classifiers/regressors are **shared** across all levels

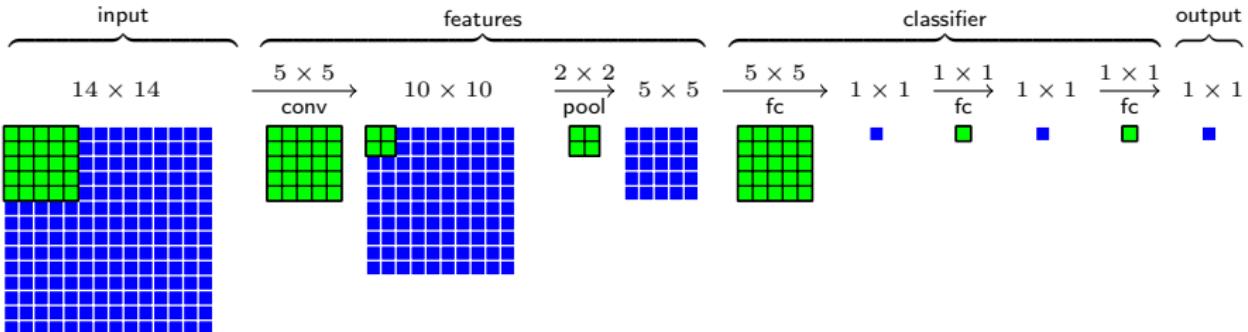
# one-stage detection

# OverFeat\*

[Sermanet et al. 2014]

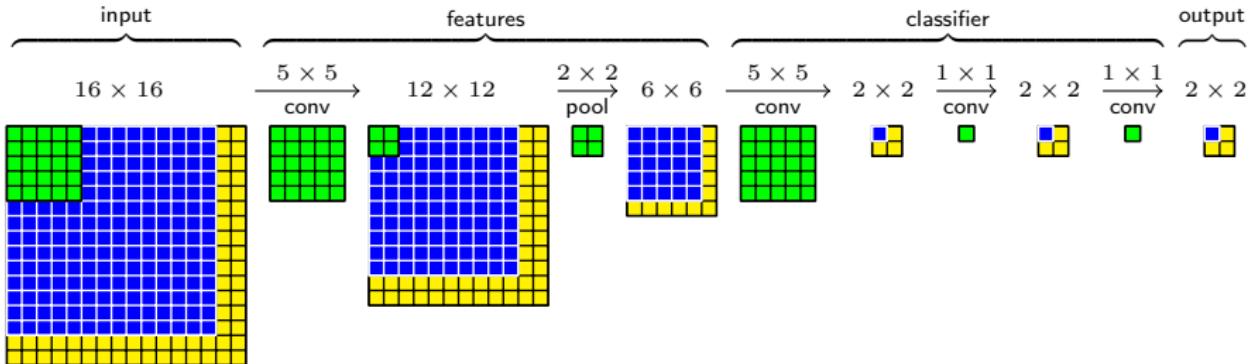
- won the ILSVRC2013 localization competition
- applied a classifier with fully connected layers densely as convolution, allowing region classification without cropping and warping
- increased output resolution with dilated convolution
- merged predictions instead of non-maxima suppression

# fully connected as convolutional



- a convolutional network with a fully connected classifier produces only one spatial output
- when applied densely over a bigger input image, it produces a spatial  $2 \times 2$  output map
- since all layers are applied convolutionally, only the yellow region needs to be recomputed

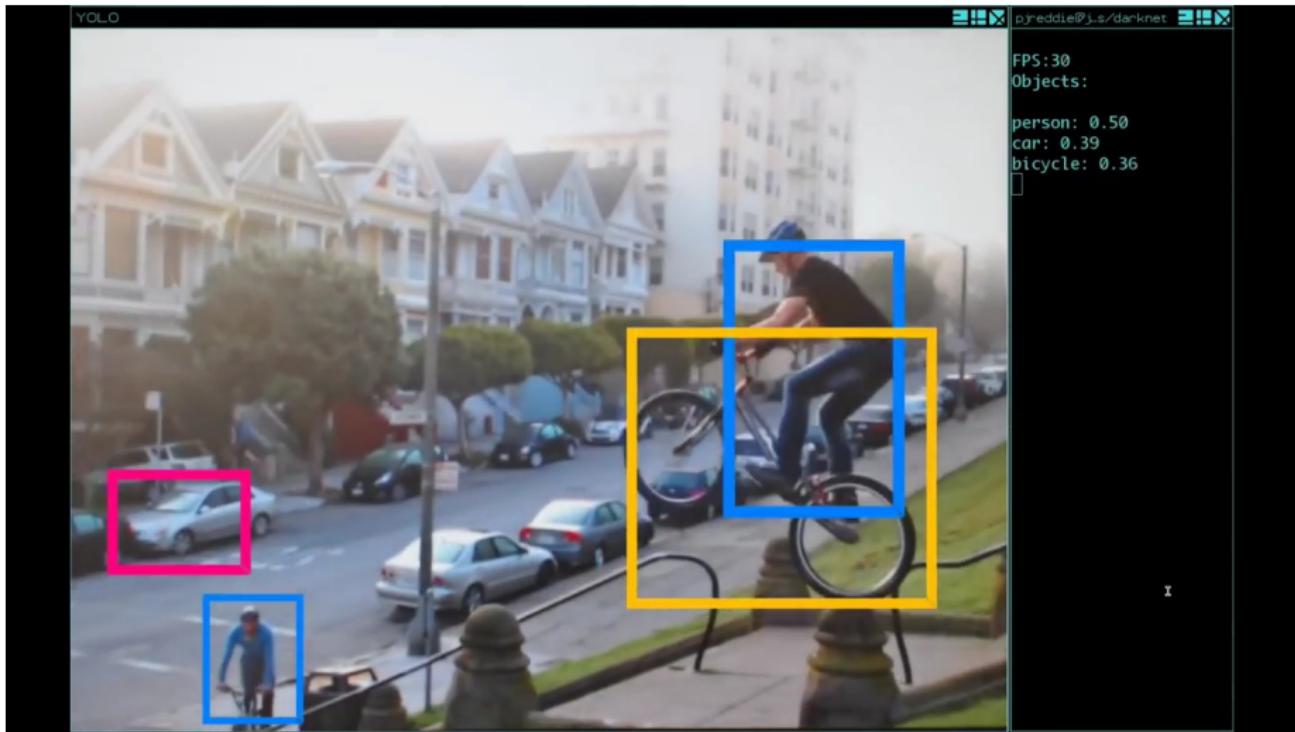
## fully connected as convolutional



- a convolutional network with a fully connected classifier produces only one spatial output
- when applied densely over a bigger input image, it produces a spatial  $2 \times 2$  output map
- since all layers are applied convolutionally, only the yellow region needs to be recomputed

# “you only look once” (YOLO)

[Redmon et al. 2016]

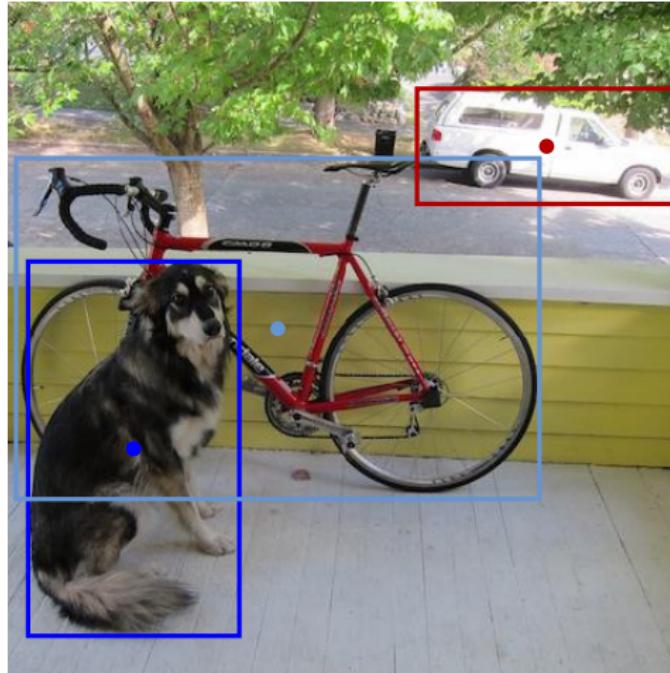


# “you only look once” (YOLO)



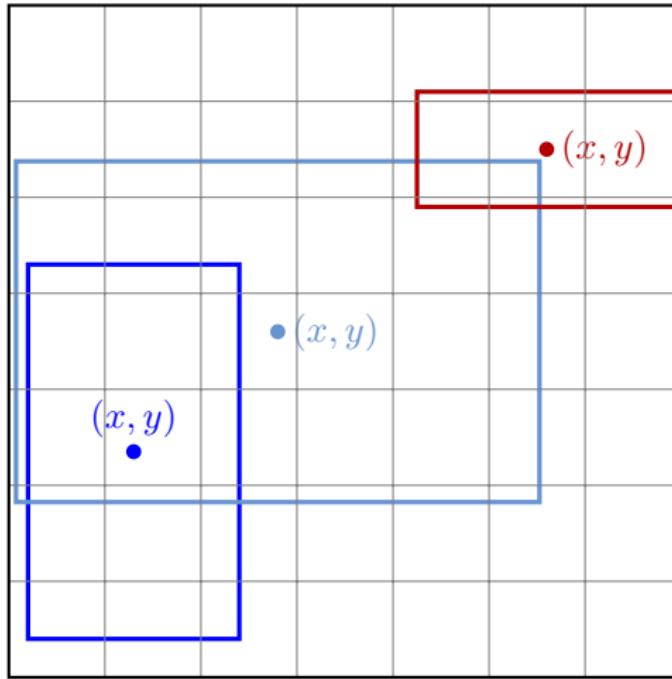
- input image

# “you only look once” (YOLO)



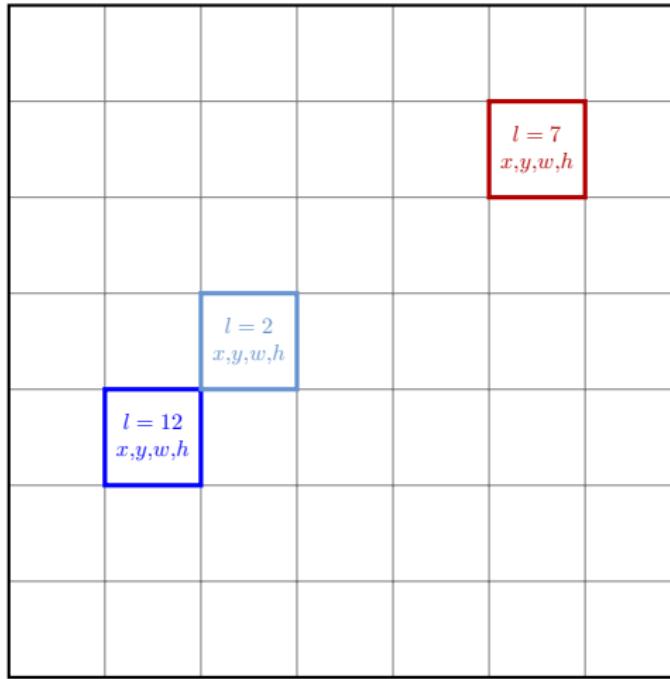
- ground truth bounding boxes and their centers

# “you only look once” (YOLO)



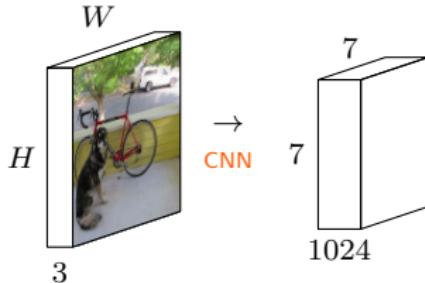
- image partitioned into  $7 \times 7$  grid and center coordinates assigned to cells

# “you only look once” (YOLO)



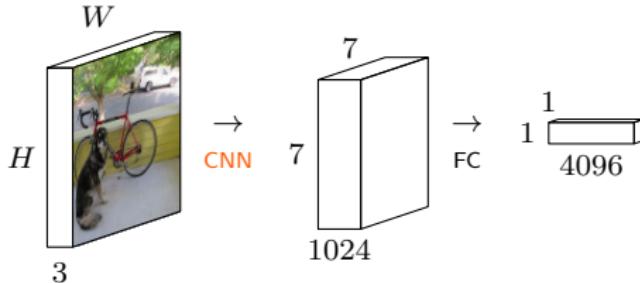
- network learns to predict up to one object per cell, including class label  $l$ , center coordinates  $x, y$  and bounding box size  $w, h$

# “you only look once” (YOLO)



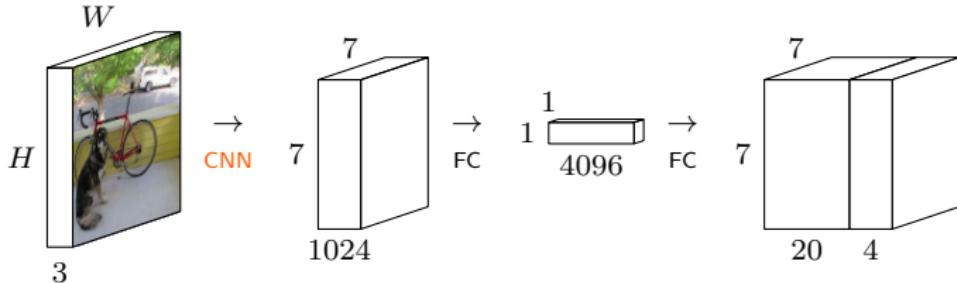
- 3-channel input  $W = H = 448$ , 24-layer NiN-like network
- fully connected layer, increasing to 4096 features
- $c = 20$  class scores and 4 bounding box coordinates at each position
- in a single stage, network performs regression from the image to a  $7 \times 7 \times 24$  tensor encoding detected classes and positions
- regression ( $\ell_2$ ) loss on both class scores and coordinates
- “objectness” score makes it look like two-stage

# “you only look once” (YOLO)



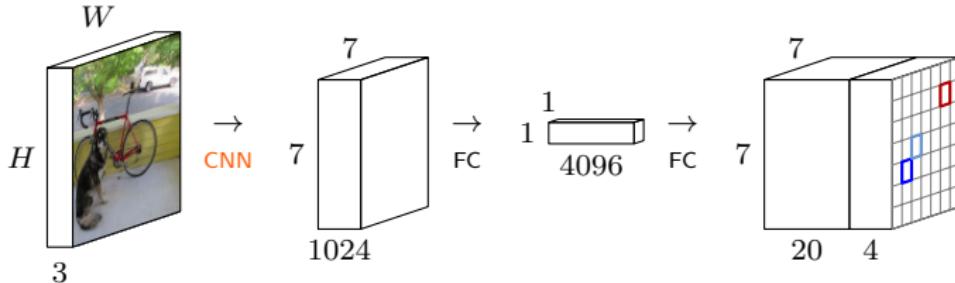
- 3-channel input  $W = H = 448$ , 24-layer NiN-like network
- fully connected layer, increasing to 4096 features
- $c = 20$  class scores and 4 bounding box coordinates at each position
- in a single stage, network performs regression from the image to a  $7 \times 7 \times 24$  tensor encoding detected classes and positions
- regression ( $\ell_2$ ) loss on both class scores and coordinates
- “objectness” score makes it look like two-stage

# “you only look once” (YOLO)



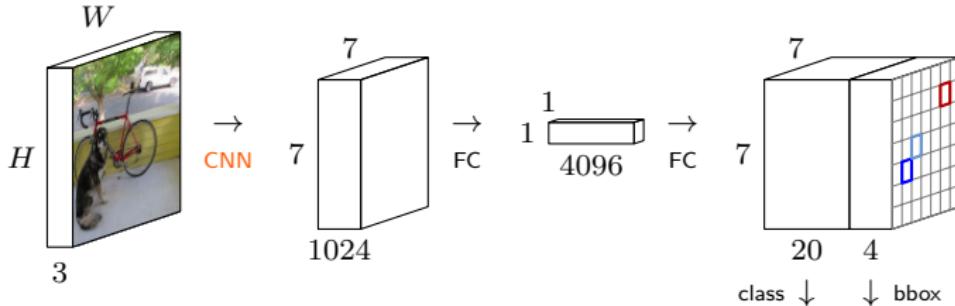
- 3-channel input  $W = H = 448$ , 24-layer NiN-like network
- fully connected layer, increasing to 4096 features
- $c = 20$  class scores and 4 bounding box coordinates at each position
- in a single stage, network performs regression from the image to a  $7 \times 7 \times 24$  tensor encoding detected classes and positions
- regression ( $\ell_2$ ) loss on both class scores and coordinates
- “objectness” score makes it look like two-stage

# “you only look once” (YOLO)



- 3-channel input  $W = H = 448$ , 24-layer NiN-like network
- fully connected layer, increasing to 4096 features
- $c = 20$  class scores and 4 bounding box coordinates at each position
- in a single stage, network performs regression from the image to a  $7 \times 7 \times 24$  tensor encoding detected classes and positions
- regression ( $\ell_2$ ) loss on both class scores and coordinates
- “objectness” score makes it look like two-stage

# “you only look once” (YOLO)



- 3-channel input  $W = H = 448$ , 24-layer NiN-like network
- fully connected layer, increasing to 4096 features
- $c = 20$  class scores and 4 bounding box coordinates at each position
- in a single stage, network performs regression from the image to a  $7 \times 7 \times 24$  tensor encoding detected classes and positions
- regression ( $\ell_2$ ) loss on both class scores and coordinates
- “objectness” score makes it look like two-stage

# “you only look once” (YOLO)

## pros

- **extremely fast**: 45fps;  $93\times$  to  $500\times$  test speedup vs. R-CNN on AlexNet, with similar performance
- end-to-end trainable, fully convolutional, one-stage detection

## cons

- only up to one prediction per cell (fixed in later versions)
- trouble localizing small objects
- low-performance compared to two-stage detectors on strong networks

# “you only look once” (YOLO)

## pros

- **extremely fast**: 45fps;  $93\times$  to  $500\times$  test speedup vs. R-CNN on AlexNet, with similar performance
- end-to-end trainable, fully convolutional, one-stage detection

## cons

- only up to one prediction per cell (fixed in later versions)
- trouble localizing small objects
- low-performance compared to two-stage detectors on strong networks

# single shot detector (SSD)

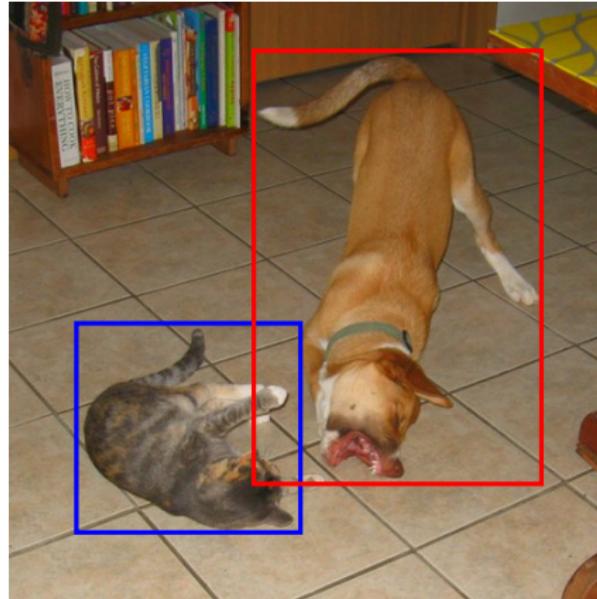
[Liu et al. 2016]



- input image

# single shot detector (SSD)

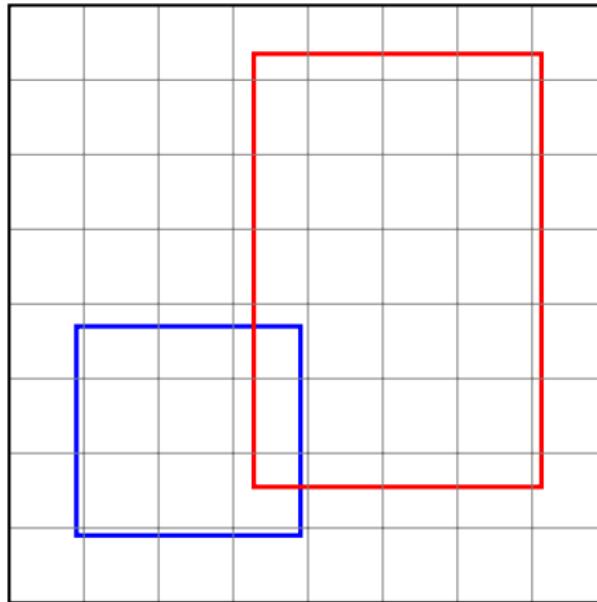
[Liu et al. 2016]



- ground truth bounding boxes

# single shot detector (SSD)

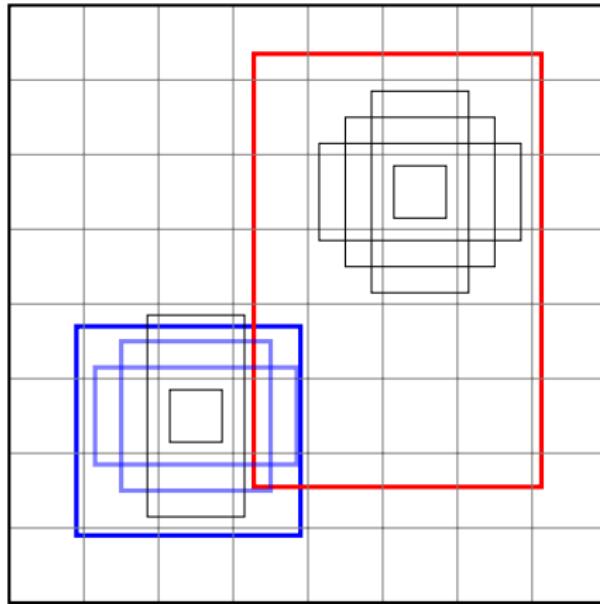
[Liu et al. 2016]



- image partitioned into  $8 \times 8$  grid

# single shot detector (SSD)

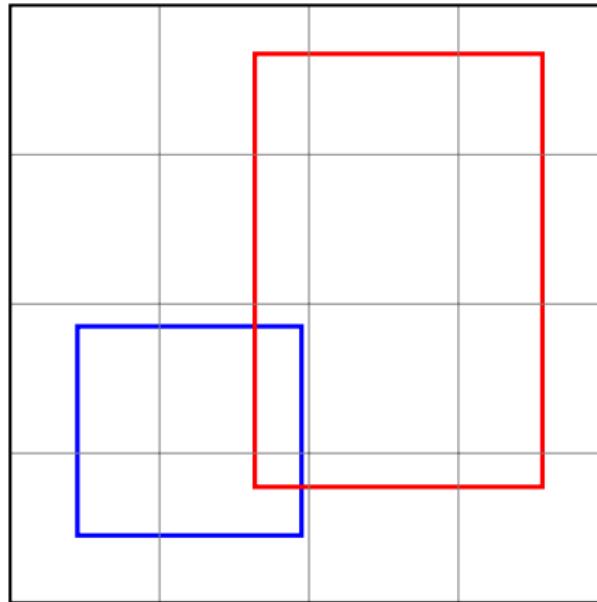
[Liu et al. 2016]



- set of anchors defined at each position, labeled as positive based on overlap with ground truth

# single shot detector (SSD)

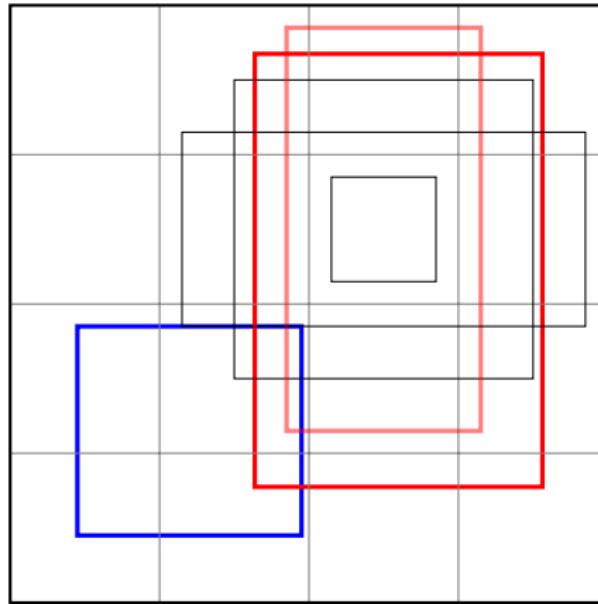
[Liu et al. 2016]



- same process at different scales, e.g.  $4 \times 4$  grid

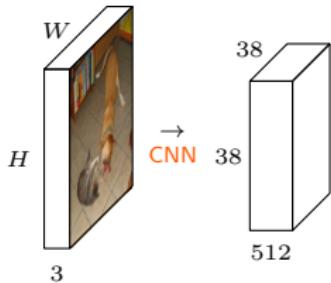
# single shot detector (SSD)

[Liu et al. 2016]



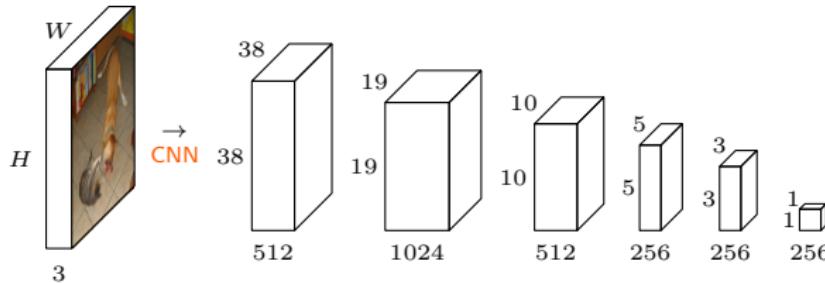
- anchor size is relative to feature map scale

# single shot detector (SSD)



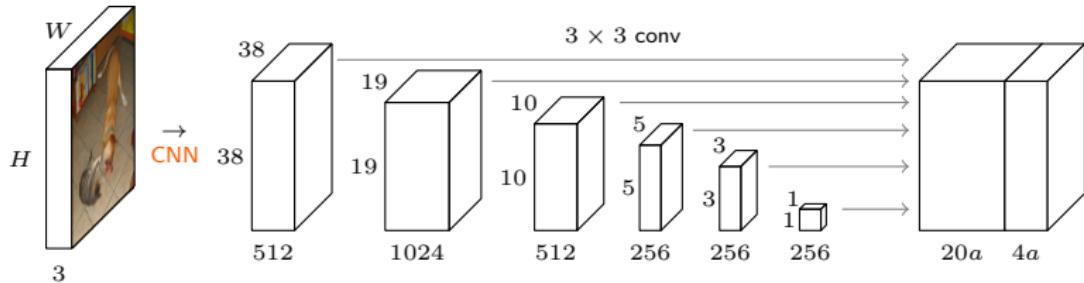
- 3-channel input  $W = H = 300$ , VGG-16 conv4-3 features
- multiple scales by convolutional layers with stride 2
- $c = 20$  classification scores and 4 bounding box coordinates relative to each of  $a = 6$  anchors at each position from each of 6 last layers: 7308 predictions per class
- softmax on scores, regression loss on coordinates

# single shot detector (SSD)



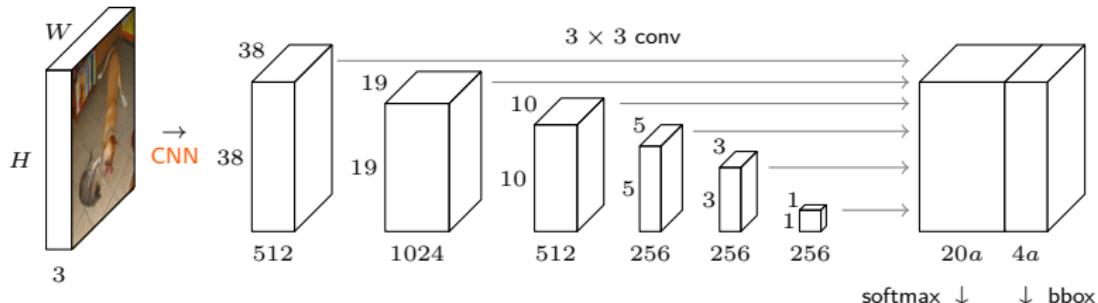
- 3-channel input  $W = H = 300$ , VGG-16 conv4-3 features
- multiple scales by convolutional layers with stride 2
- $c = 20$  classification scores and 4 bounding box coordinates relative to each of  $a = 6$  anchors at each position from each of 6 last layers: 7308 predictions per class
- softmax on scores, regression loss on coordinates

# single shot detector (SSD)



- 3-channel input  $W = H = 300$ , VGG-16 conv4-3 features
- multiple scales by convolutional layers with stride 2
- $c = 20$  classification scores and 4 bounding box coordinates relative to each of  $a = 6$  anchors at each position from each of 6 last layers: 7308 predictions per class
- softmax on scores, regression loss on coordinates

# single shot detector (SSD)



- 3-channel input  $W = H = 300$ , VGG-16 conv4-3 features
- multiple scales by convolutional layers with stride 2
- $c = 20$  classification scores and 4 bounding box coordinates relative to each of  $a = 6$  anchors at each position from each of 6 last layers: 7308 predictions per class
- softmax on scores, regression loss on coordinates

# single shot detector (SSD)

## pros

- best trade-off: 23 (SSD500) or 58fps (SSD300) with performance closer (or superior) to faster R-CNN rather than YOLO
- **many scales** at no extra cost: many more detections compared to YOLO, no need for RoI pooling
- bounding box regression is **convolutional** like RPN, but simpler pipeline like YOLO and more aspect ratios with same number of anchors

## cons

- pyramid starts at low resolution: difficulty with small objects

# single shot detector (SSD)

## pros

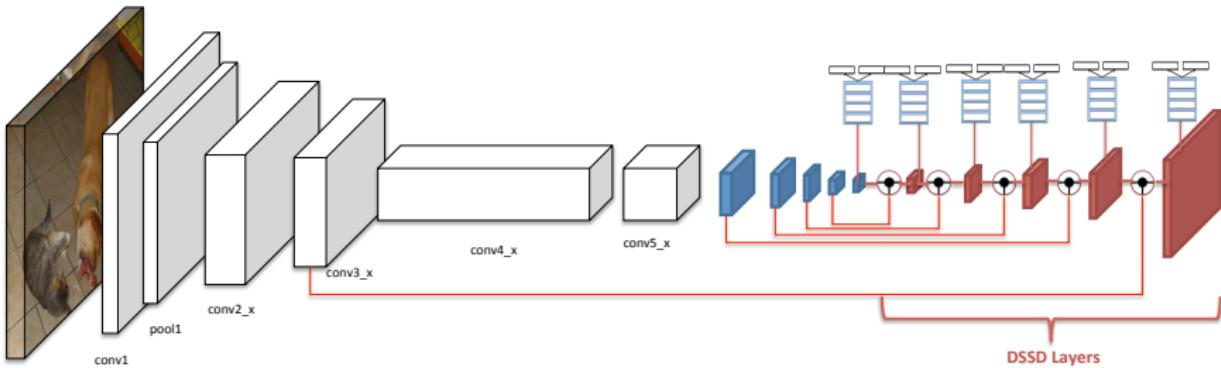
- best trade-off: 23 (SSD500) or 58fps (SSD300) with performance closer (or superior) to faster R-CNN rather than YOLO
- many scales at no extra cost: many more detections compared to YOLO, no need for RoI pooling
- bounding box regression is convolutional like RPN, but simpler pipeline like YOLO and more aspect ratios with same number of anchors

## cons

- pyramid starts at low resolution: difficulty with small objects

# deconvolutional single shot detector (DSSD)\*

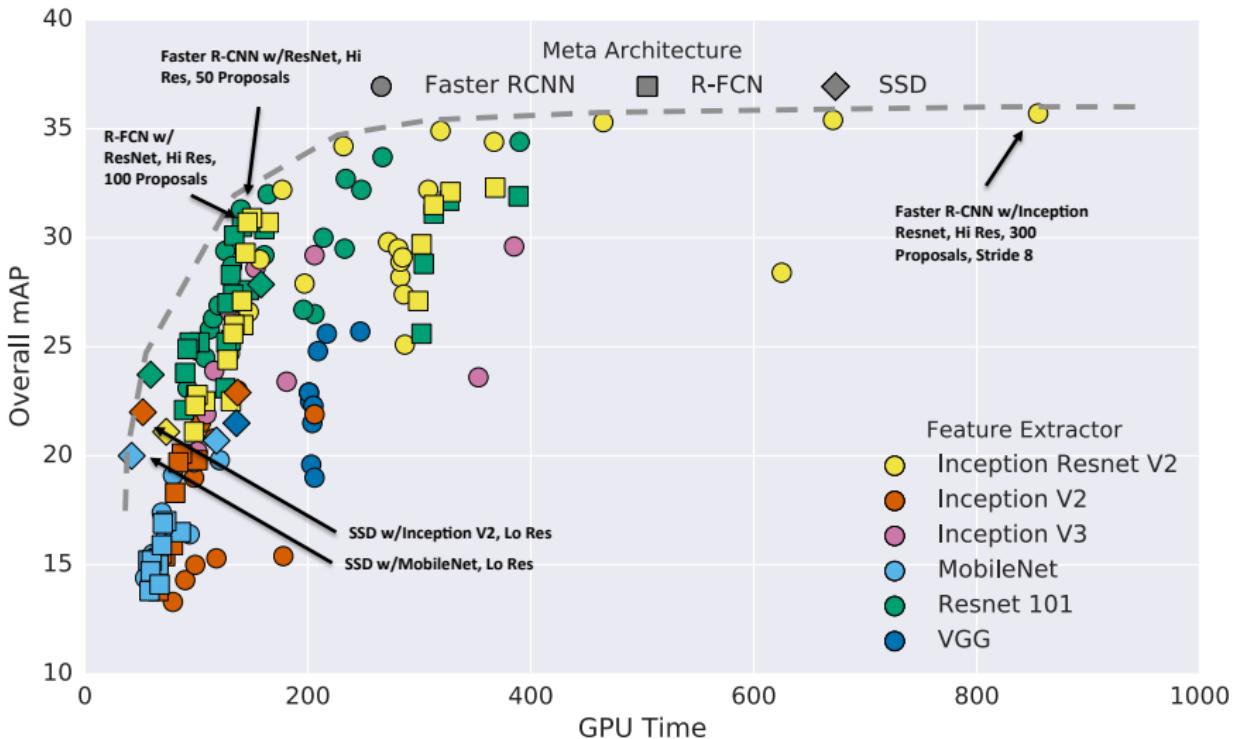
[Fu et al. 2017]



- builds on SSD on ResNet-101, introducing large-scale context
- similar to FPN, but one-stage:
  - deconvolution ( ) upsamples: high-resolution, high-level features
  - prediction ( ) (classifier + regressor) at all top-down layers
- improves accuracy, especially on small objects
- only slightly slower than SSD

# speed-accuracy trade-offs

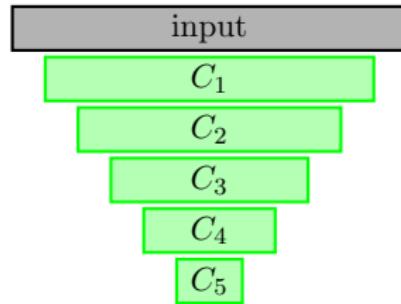
[Huang et al. 2016]



Huang, Rathod, Sun, Zhu, Korattikara, Fathi, Fischer, Wojna, Song, Guardarrama and Murphy 2016. Speed-Accuracy Trade-Offs for Modern Convolutional Object Detectors.

# RetinaNet

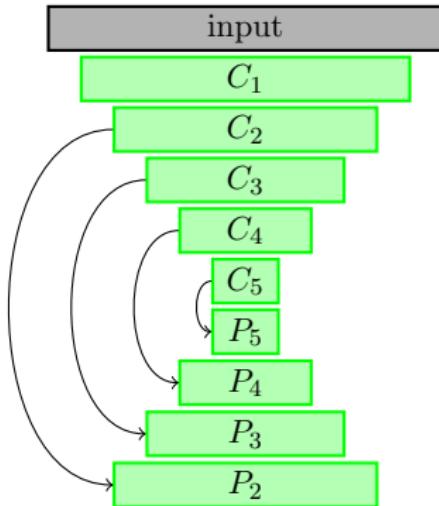
[Lin et al. 2017]



- base network: ResNet-101
- feature pyramid network
- multi-scale dense detection

# RetinaNet

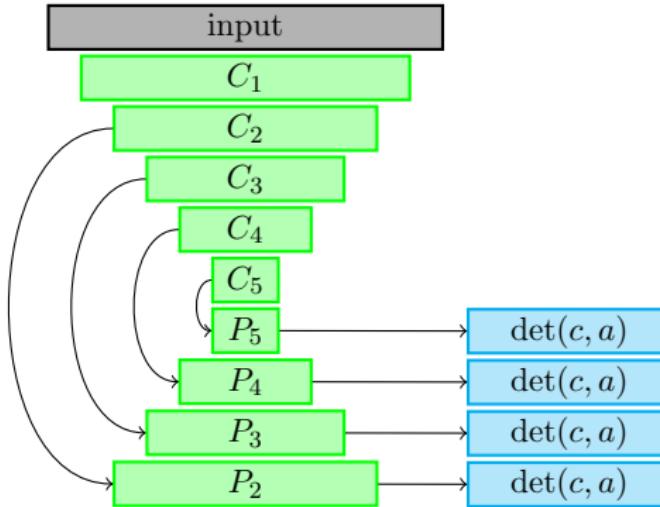
[Lin et al. 2017]



- base network: ResNet-101
- feature pyramid network
- multi-scale dense detection

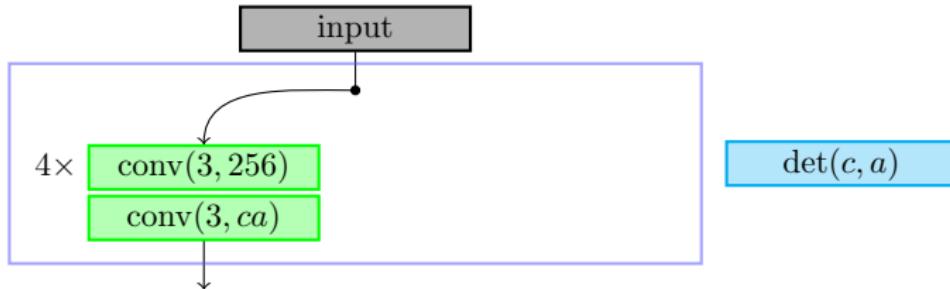
# RetinaNet

[Lin et al. 2017]



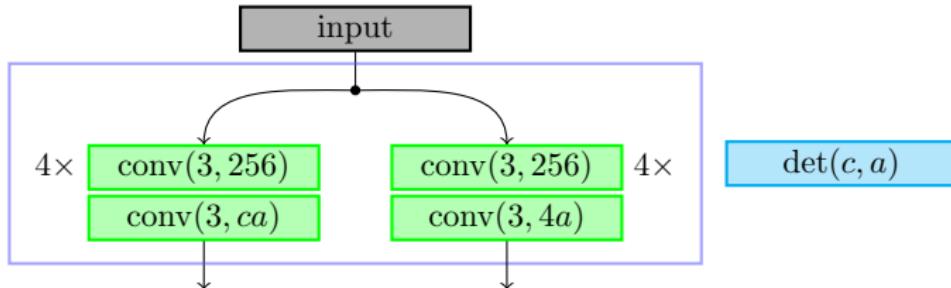
- base network: ResNet-101
- feature pyramid network
- multi-scale dense detection

# RetinaNet: dense detection



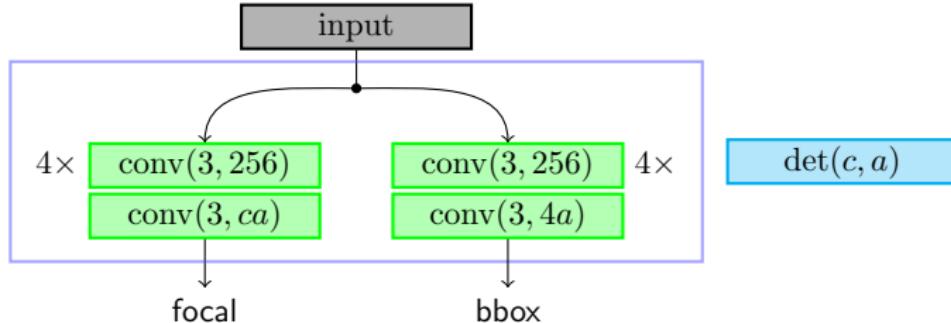
- $c$  classification scores for each of  $a = 9$  anchors at each position (3 scales, 3 aspect ratios)
- 4 bounding box coordinates relative to each anchor at each position
- **focal loss** on class scores, regression loss on coordinates
- no parameters shared between classification and regression branches
- parameters of detection subnets shared across all pyramid levels

# RetinaNet: dense detection



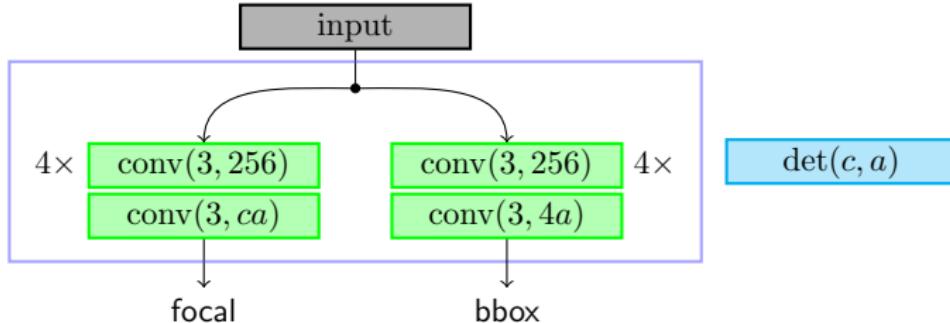
- $c$  classification scores for each of  $a = 9$  anchors at each position (3 scales, 3 aspect ratios)
- 4 bounding box coordinates relative to each anchor at each position
- focal loss on class scores, regression loss on coordinates
- no parameters shared between classification and regression branches
- parameters of detection subnets shared across all pyramid levels

# RetinaNet: dense detection



- $c$  classification scores for each of  $a = 9$  anchors at each position (3 scales, 3 aspect ratios)
- 4 bounding box coordinates relative to each anchor at each position
- **focal loss** on class scores, regression loss on coordinates
- no parameters shared between classification and regression branches
- parameters of detection subnets shared across all pyramid levels

# RetinaNet: dense detection



- $c$  classification scores for each of  $a = 9$  anchors at each position (3 scales, 3 aspect ratios)
- 4 bounding box coordinates relative to each anchor at each position
- **focal loss** on class scores, regression loss on coordinates
- no parameters shared between classification and regression branches
- parameters of detection subnets shared across all pyramid levels

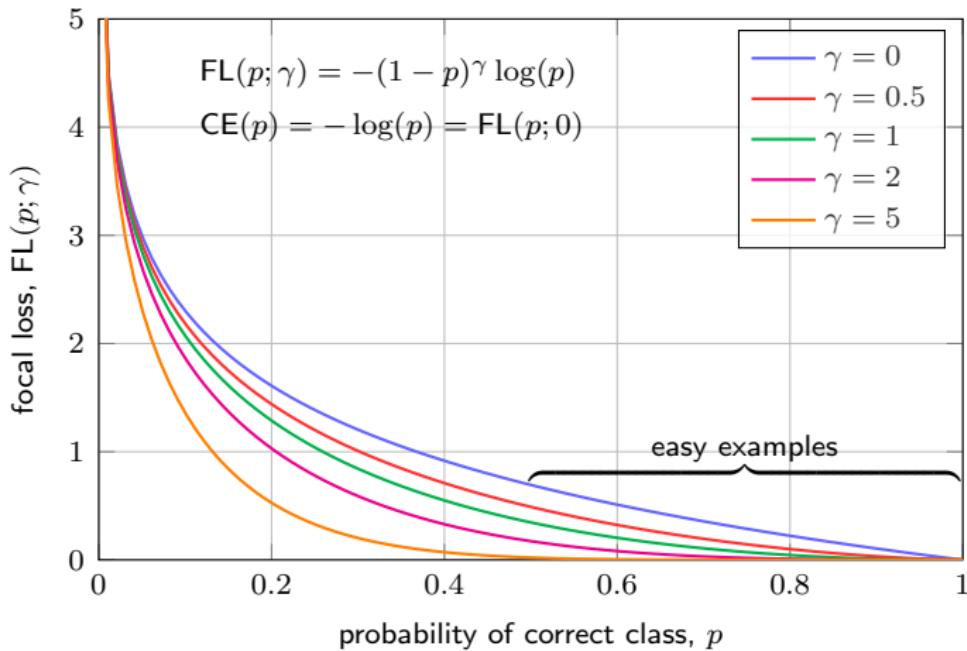
# what is wrong with dense detection?

- in a two-stage detector, the classifier is applied to a **sparse** set of candidate object locations, which are found by binary classification (object/non-object)
- in a one-stage detector, the classifier is applied to a **dense** set of locations (e.g. a regular grid), which introduces **extreme class imbalance** between foreground-background
- there is a vast number of **easy negatives** that can overwhelm the detector
- as an alternative to OHEM, design the loss function such that it does not penalize well-classified examples

# what is wrong with dense detection?

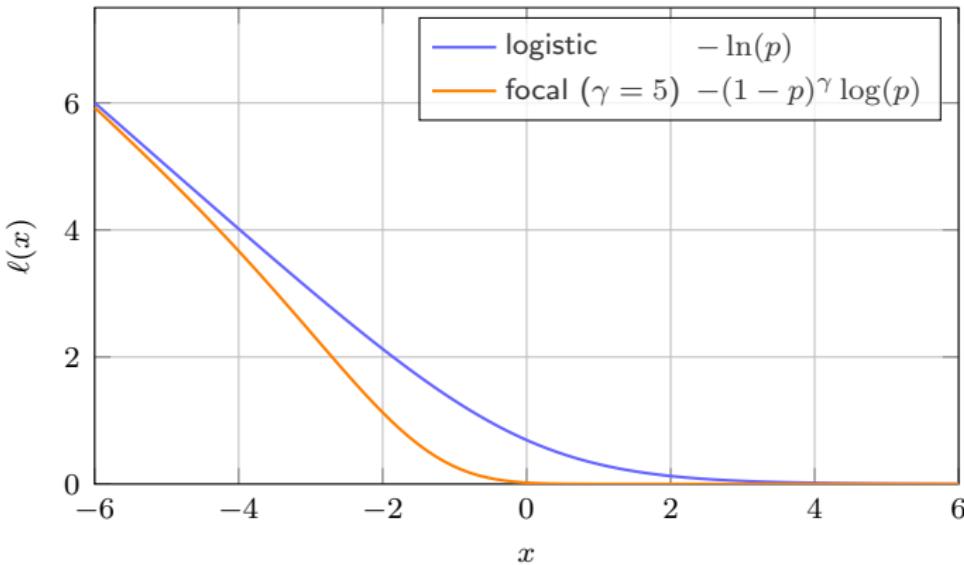
- in a two-stage detector, the classifier is applied to a **sparse** set of candidate object locations, which are found by binary classification (object/non-object)
- in a one-stage detector, the classifier is applied to a **dense** set of locations (e.g. a regular grid), which introduces **extreme class imbalance** between foreground-background
- there is a vast number of **easy negatives** that can overwhelm the detector
- as an alternative to OHEM, design the loss function such that it does not penalize well-classified examples

# focal loss



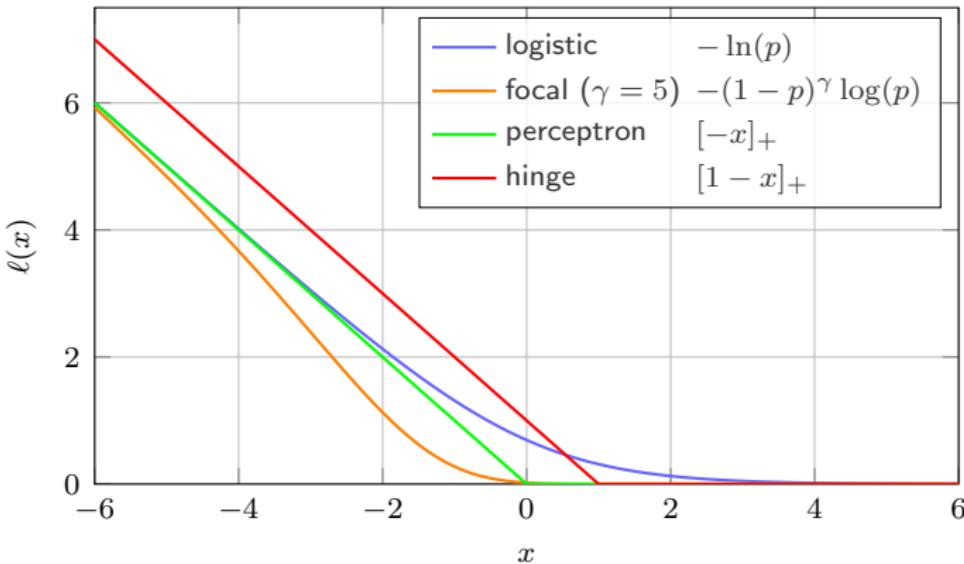
- reduces the relative loss for well-classified examples ( $p > 0.5$ ), putting more focus on hard, misclassified examples

# remember the perceptron loss? the margin?



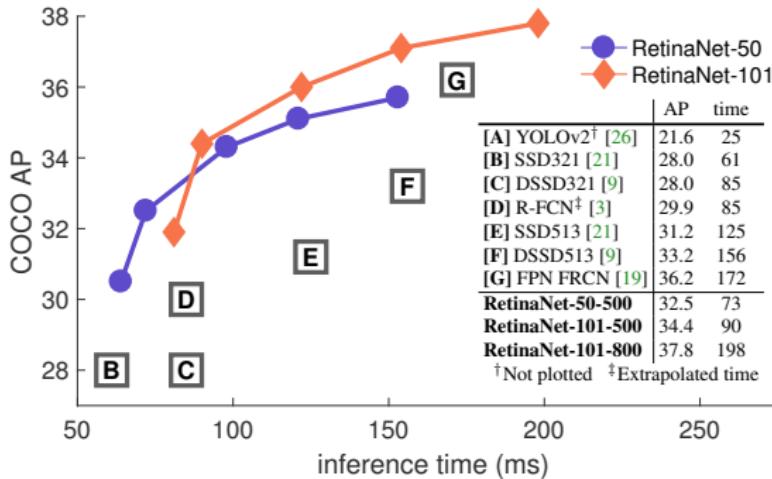
- the probability of the correct class is  $p = \sigma(x) = \frac{1}{1+e^{-x}}$ , where  $x = sa$ ,  $s \in \{-1, 1\}$  is the “sign” target variable, and  $a$  the activation
- easy example means  $p > 0.5$ , or  $x > 0$
- perceptron loss is zero for such examples; logistic and hinge are not

# remember the perceptron loss? the margin?



- the probability of the correct class is  $p = \sigma(x) = \frac{1}{1+e^{-x}}$ , where  $x = sa$ ,  $s \in \{-1, 1\}$  is the “sign” target variable, and  $a$  the activation
- easy example means  $p > 0.5$ , or  $x > 0$
- perceptron loss is zero for such examples; logistic and hinge are not

# RetinaNet: performance



- RetinaNet on ResNet-50-FPN and ResNet-101-FPN performance on COCO at five scales (400-800 pixels)
- outperforms all one-stage and two-stage detectors

## one-stage vs. two-stage

- two-stage fights **class imbalance**; alternatively, use batch sampling, hard negative mining, or a better loss function
- two-stage defines regions at different **scales**; alternatively, use multiple scales from a feature pyramid
- two-stage resamples regions at different **aspect ratios**, or with **deformable parts**; this has not been explored with feature pyramids or one-stage detectors yet

# attention networks\*



- of course, there can be more stages!
- AttentionNet iterates bounding box regression and classification

## summary

- **background**: detectors (Viola & Jones, DPM, ISM, ESS), object proposals, NMS, evaluation
- **two-stage** detection: R-CNN, SPP, fast/faster R-CNN, RPN
- **parts**: R-FCN, spatial transformers, deformable convolution
- **upsampling\***: FCN, feature pyramids, TDM, FPN
- **one-stage** detection: OverFeat\*, YOLO, SSD\*, DSSD\*, RetinaNet\*, focal loss
- with feature pyramids, multi-scale representation and appropriate loss, the gap between one- and two-stage detection is closing
- **attentional cascade** classifiers are developed in parallel