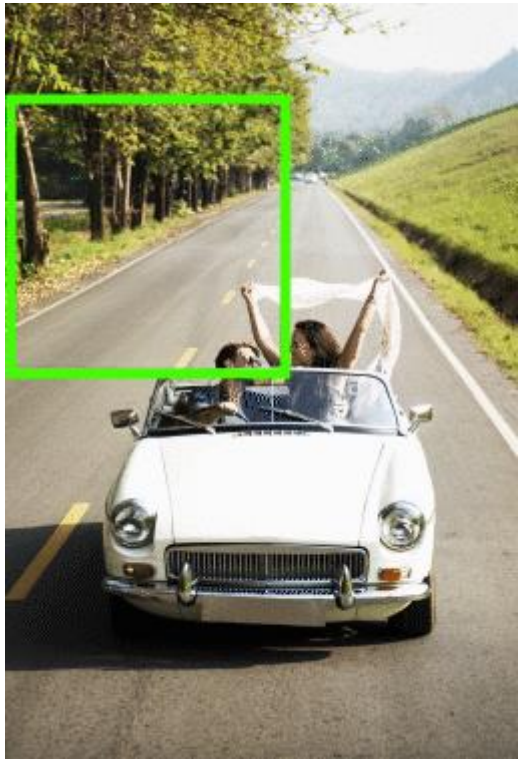
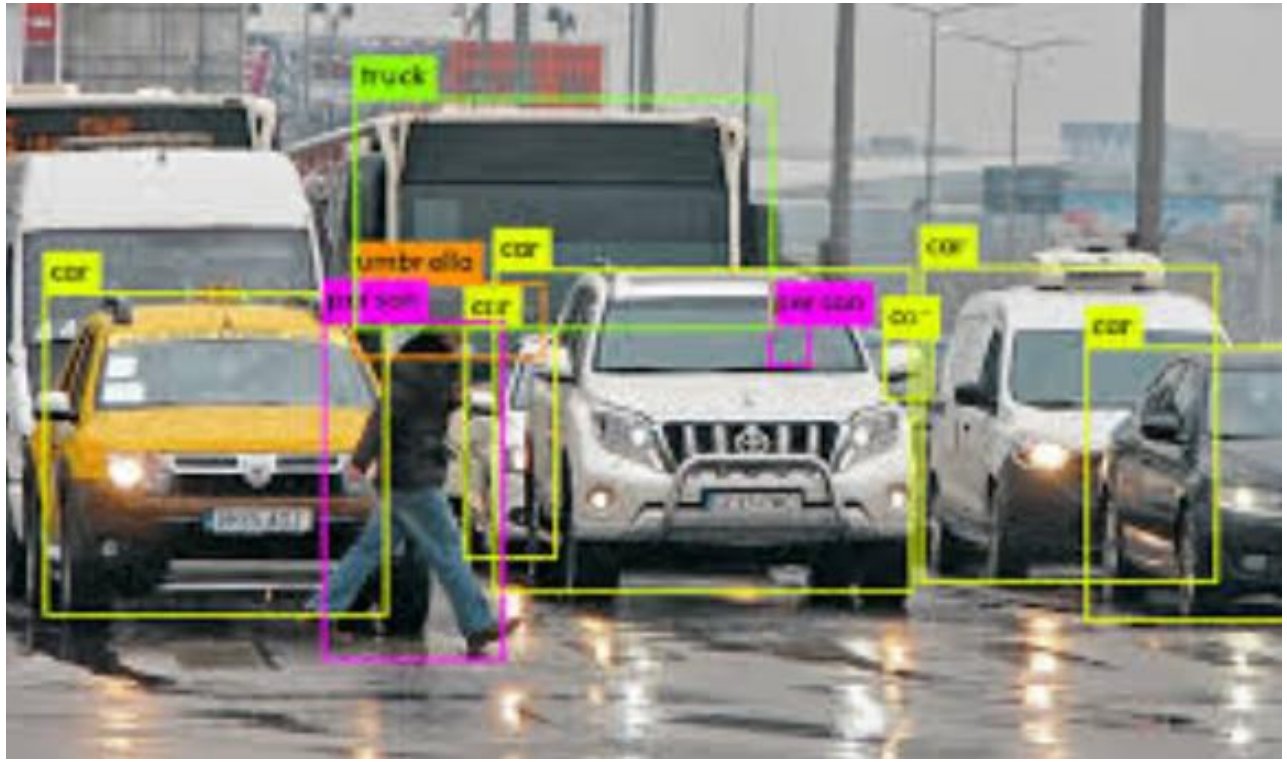
Object detection

Biplab Banerjee

The object detection problem





Datasets

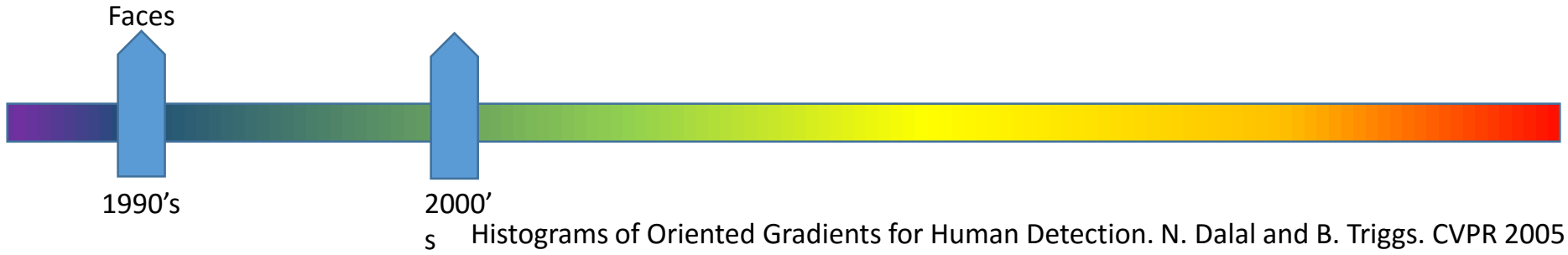
- Face detection
- One category: face
- Frontal faces
- Fairly rigid, unoccluded

1990's

Human Face Detection in Visual Scenes. H. Rowley, S. Baluja, T. Kanade. 1995.

Pedestrians

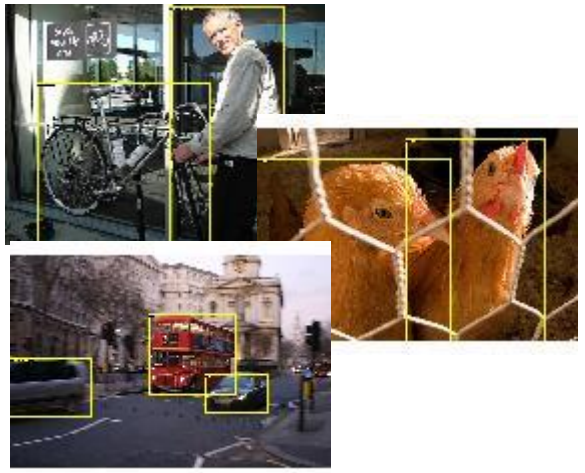
- One category: pedestrians
- Slight pose variations and small distortions
- Partial occlusions

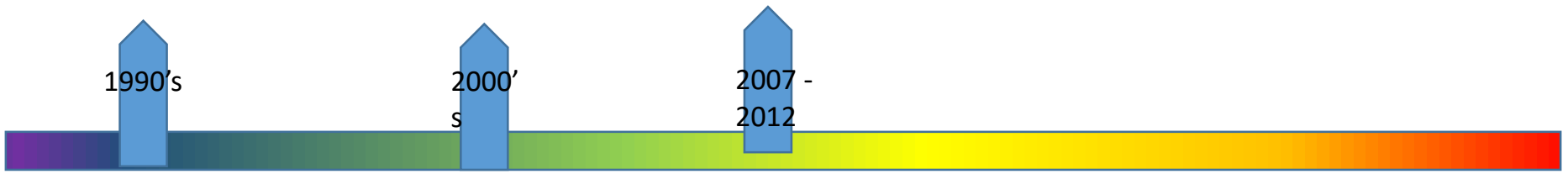


Histograms of Oriented Gradients for Human Detection. N. Dalal and B. Triggs. CVPR 2005

PASCAL VOC

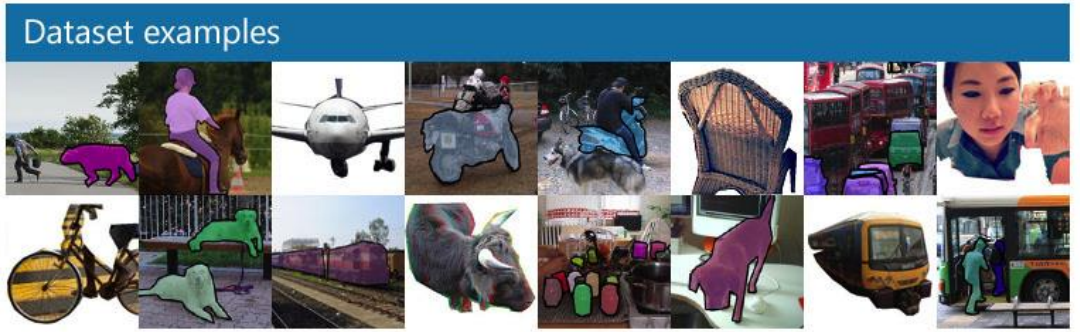
- 20 categories
- 10K images
- Large pose variations, heavy occlusions
- Generic scenes
- Cleaned up performance metric

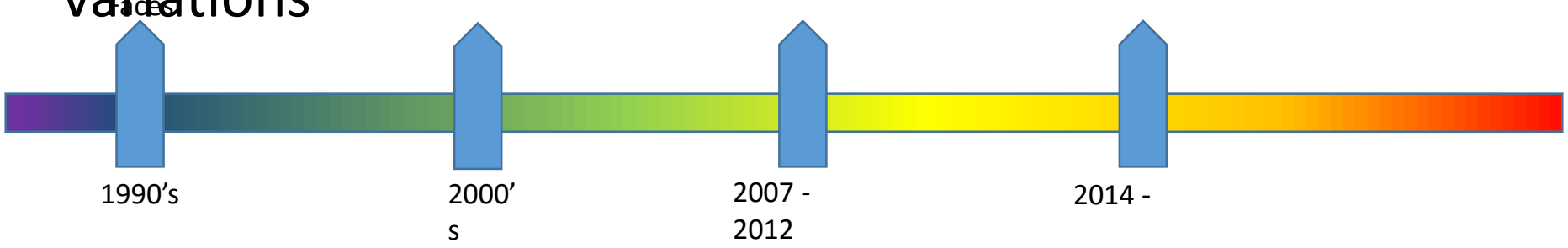




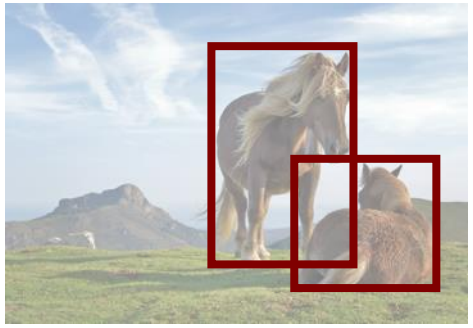
Coco

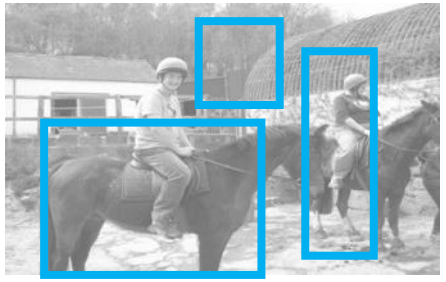
- 80 diverse categories
- 100K images
- Heavy occlusions, many objects per image, large scale variations

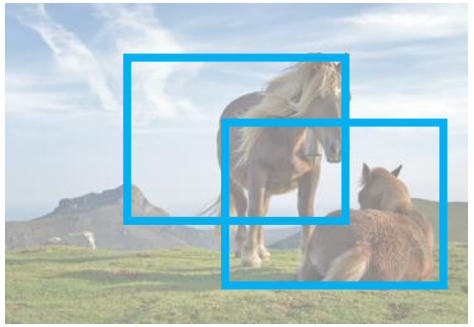




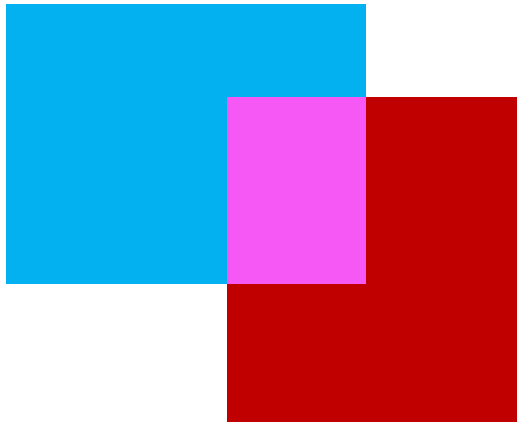
Evaluation metric







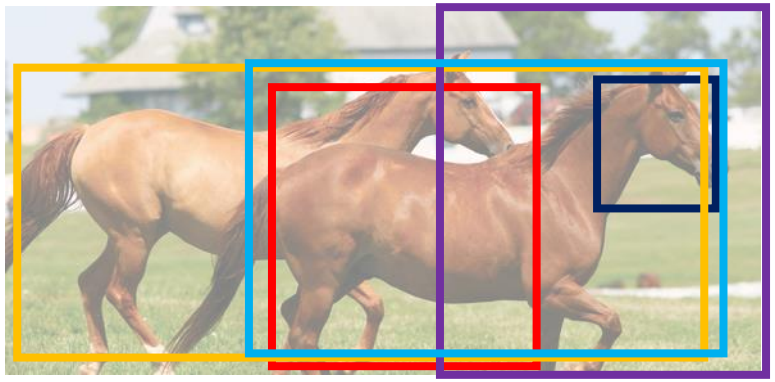
Matching detections to ground truth



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

Why is detection hard(er)?

- Precise localization



Why is detection hard(er)?

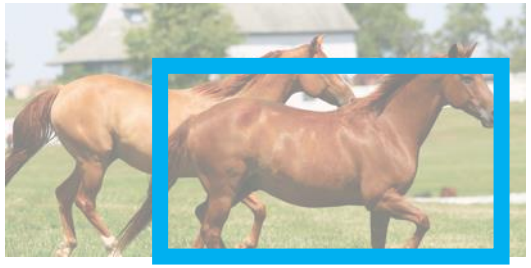
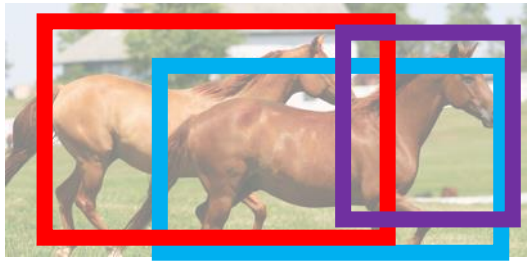
- Much larger impact of pose

Why is detection hard(er)?

- Occlusion makes localization difficult

Why is detection hard(er)?

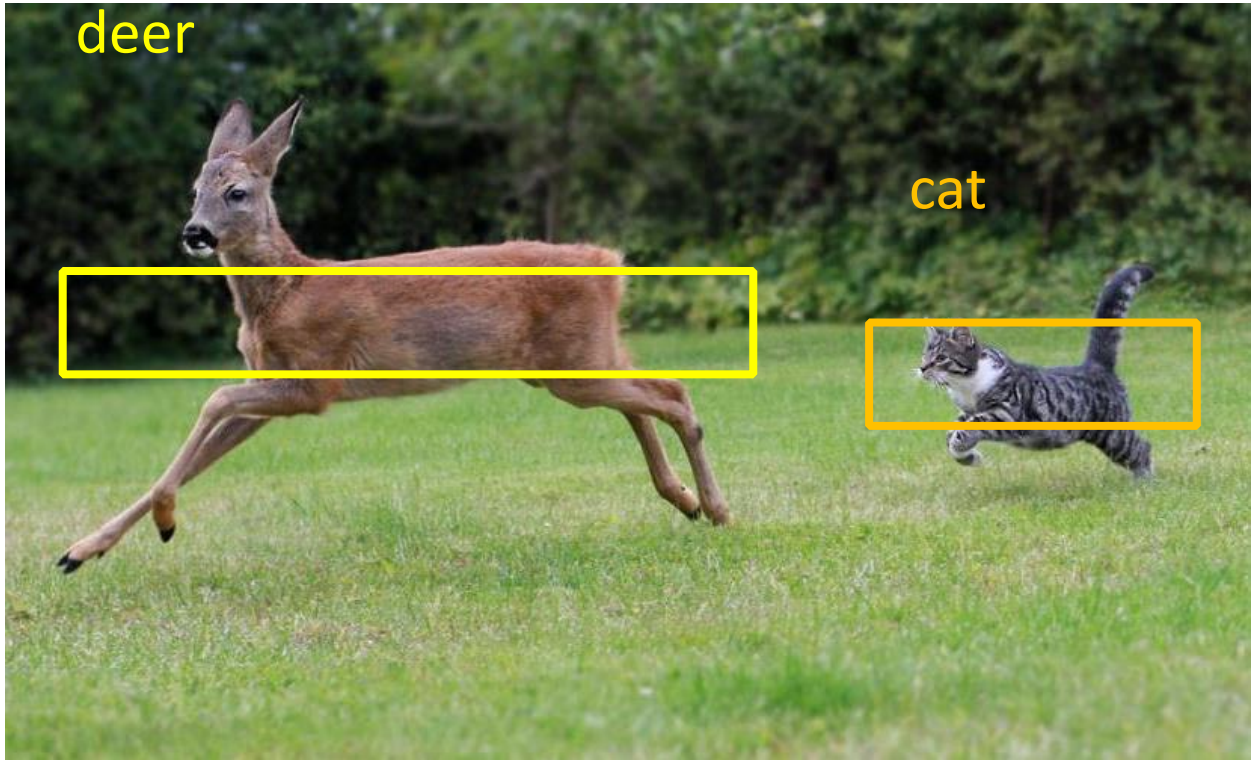
- Counting

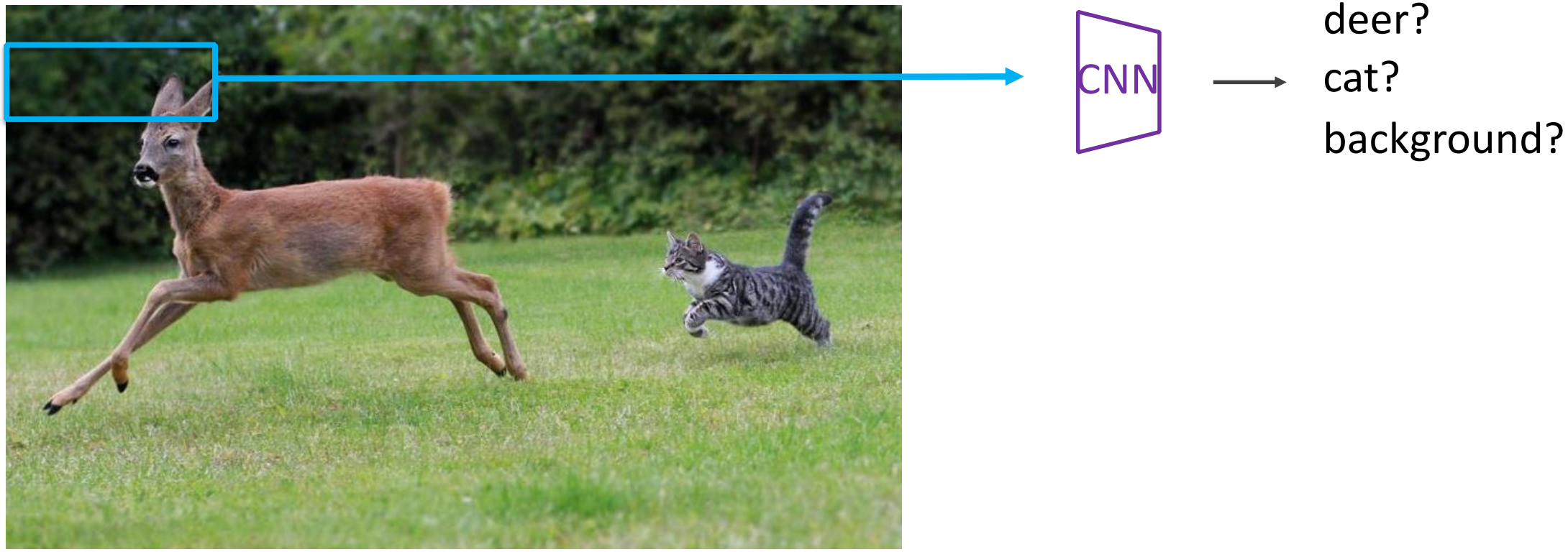

Why is detection hard(er)?

- Small objects

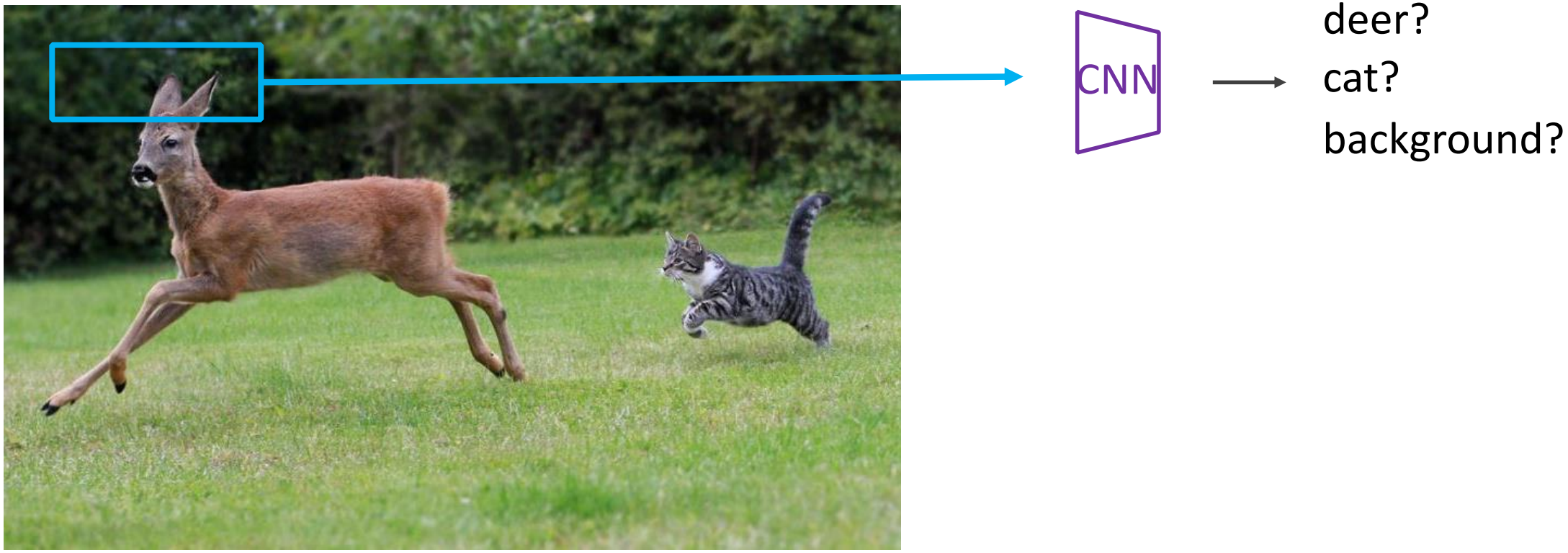
Object Detection



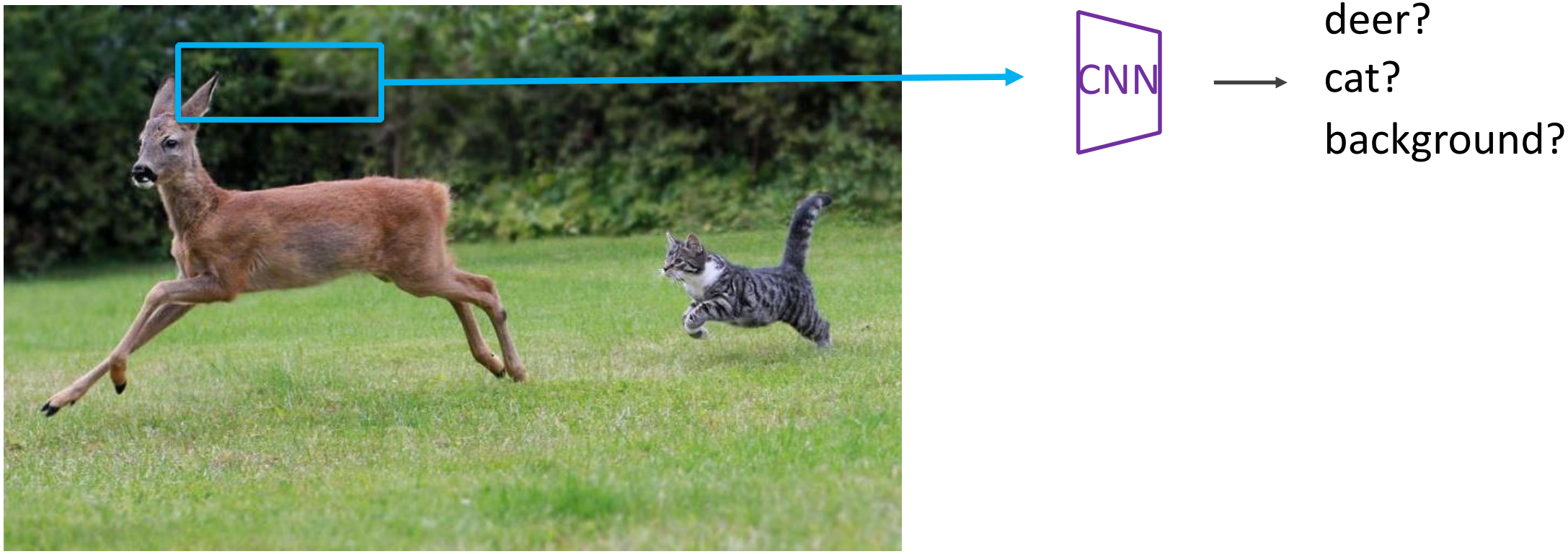
Object Detection as Classification



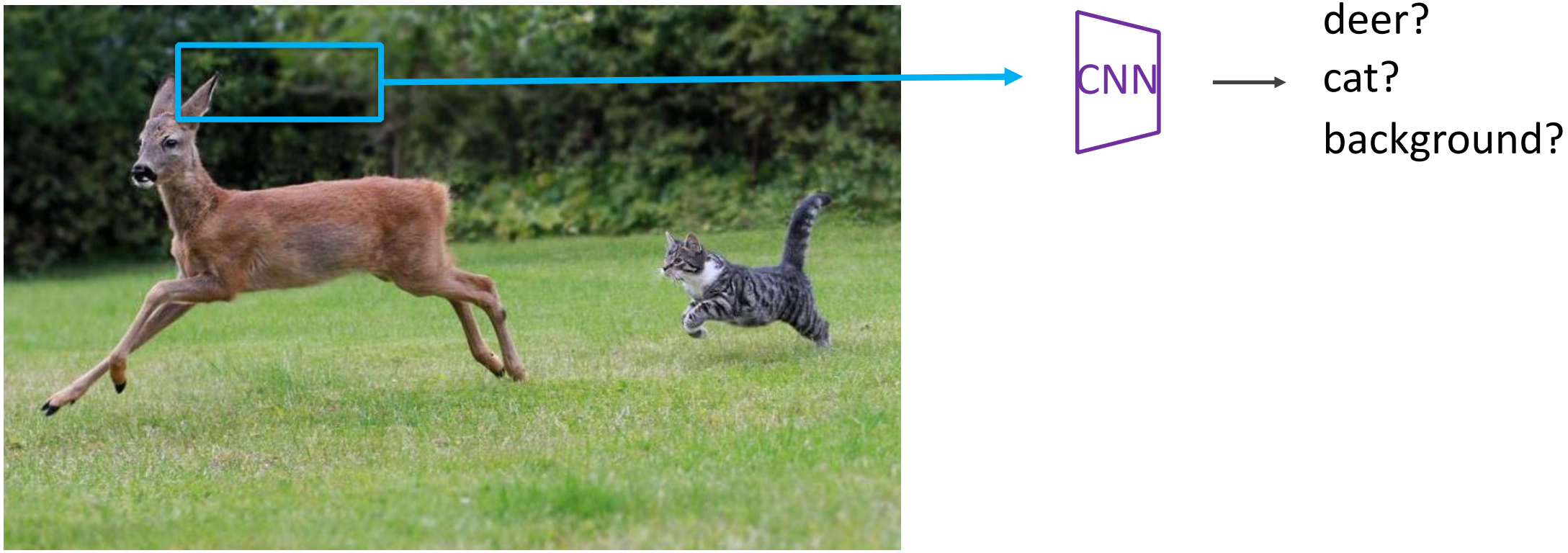
Object Detection as Classification



Object Detection as Classification



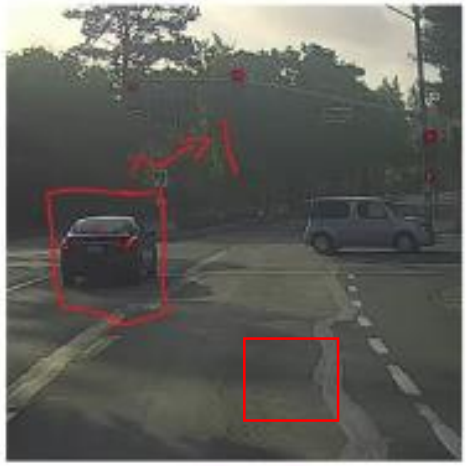
Object Detection as Classification with Sliding Window



Problems with sliding window approach

Training set:

x	y
	1
	1
	1
	0

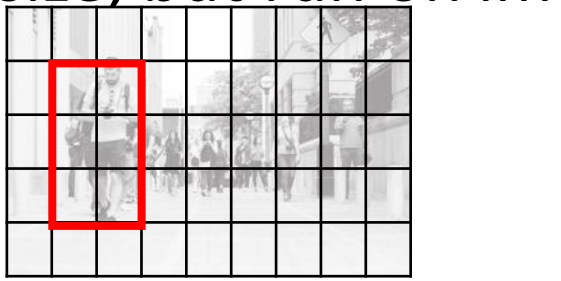


1. Fine-tune the CNN with this new training data
2. Pass the sliding windows through the CNN for binary classification
3. Huge computational cost! Can we do better?

Dealing with scale

Dealing with scale

- Use same window size, but run on *image pyramid*



Scanning window results on PASCAL

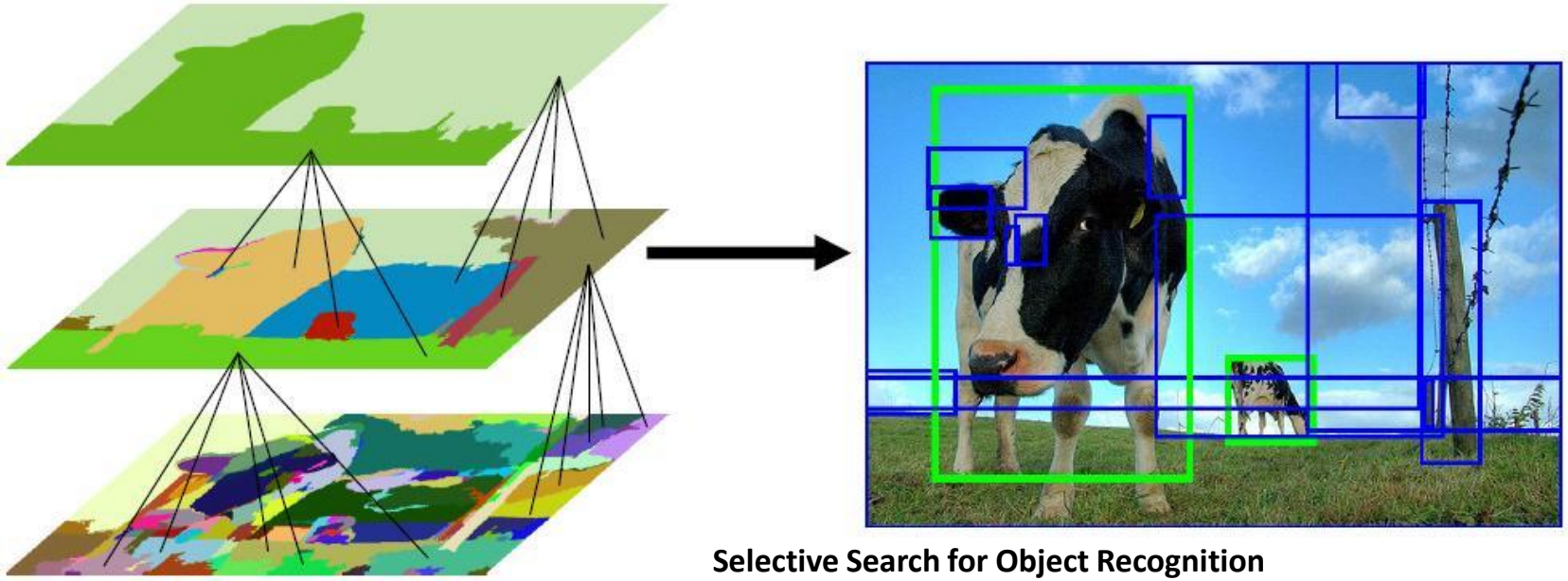
	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%

Reference systems

Slide credit : Ross Girshick

Idea 2: Object proposals

- Use segmentation to produce ~5K candidates



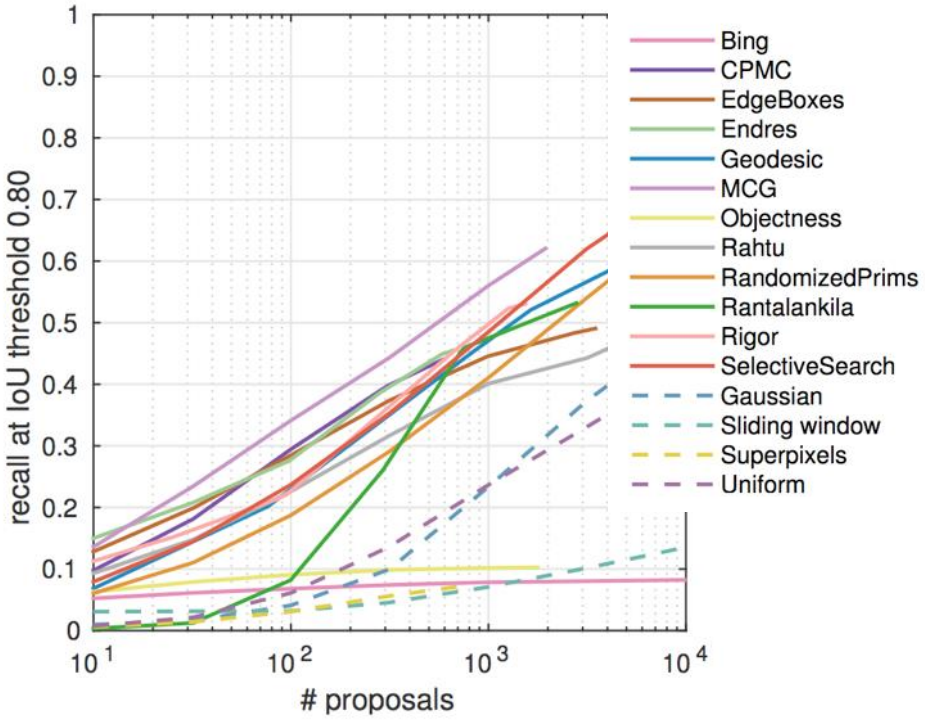
Selective Search for Object Recognition
J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, A. W. M. Smeulders
In International Journal of Computer Vision 2013.

Idea 2: object proposals

- Many different segmentation algorithms (k-means on color, k-means on color+position, N-cuts….)
- Many hyperparameters (number of clusters, weights on edges)
- Try everything!
  - Every cluster is a candidate object
  - Thousands of segmentations -> thousands of candidate objects

Idea 2: Object proposals

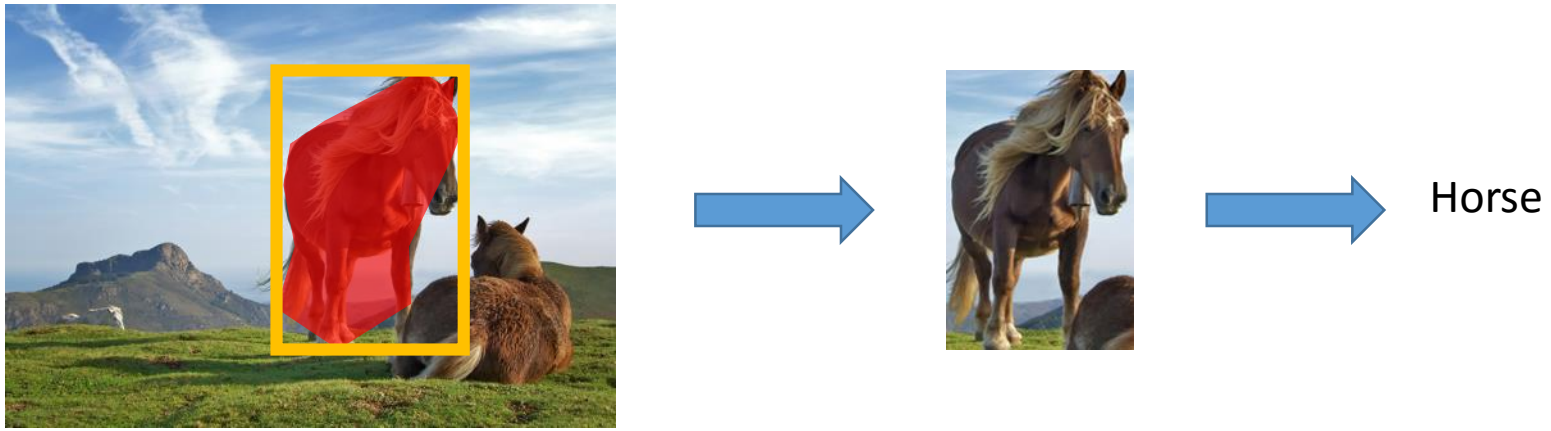
- Tens of ways of generating candidates (“proposals”)
- What fraction of ground truth objects have proposals near them?



What makes for effective detection proposals? J. Hosang, R. Benenson, P. Dollar, B. Schiele. In TPAMI

What do we do with proposals?

- Each proposal is a group of pixels
- Take tight fitting box and *classify it*
- *Can leverage any image classification approach*



Proposal methods results

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%

Reference systems

Slide credit : Ross Girshick

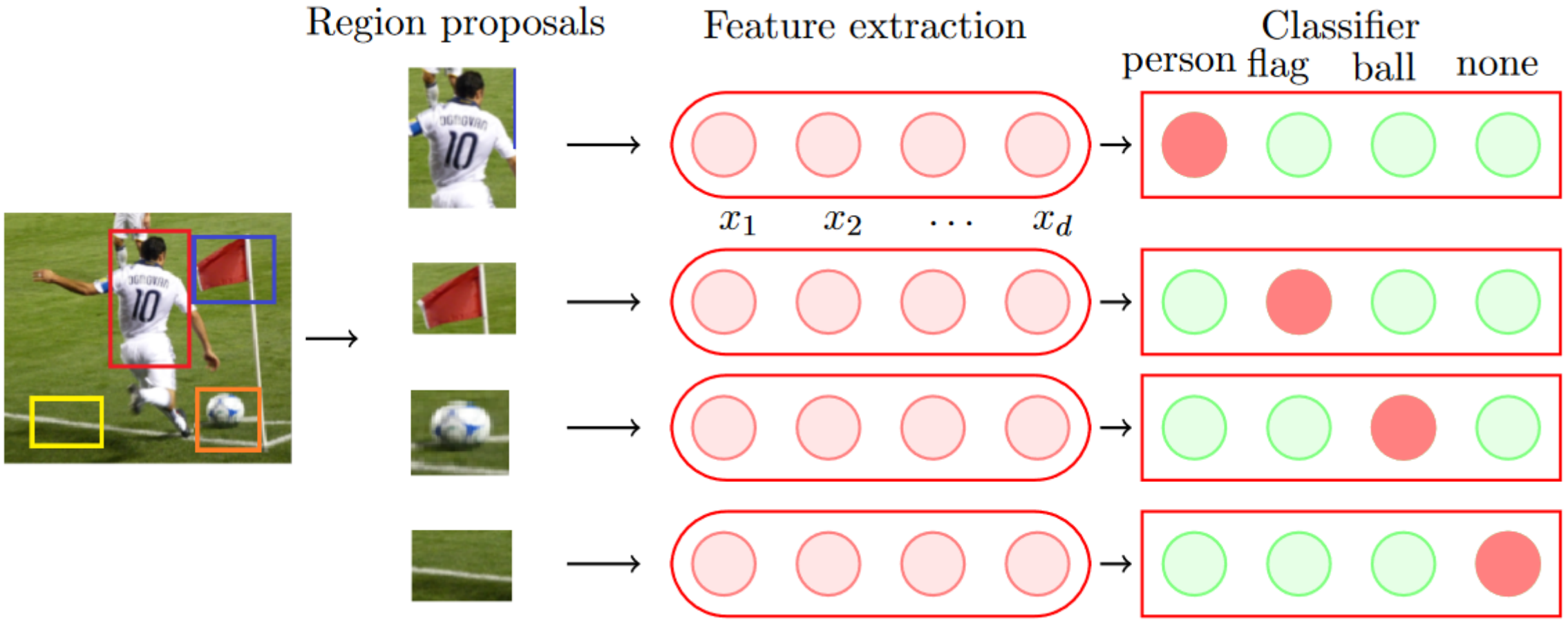
Proposal methods results

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%

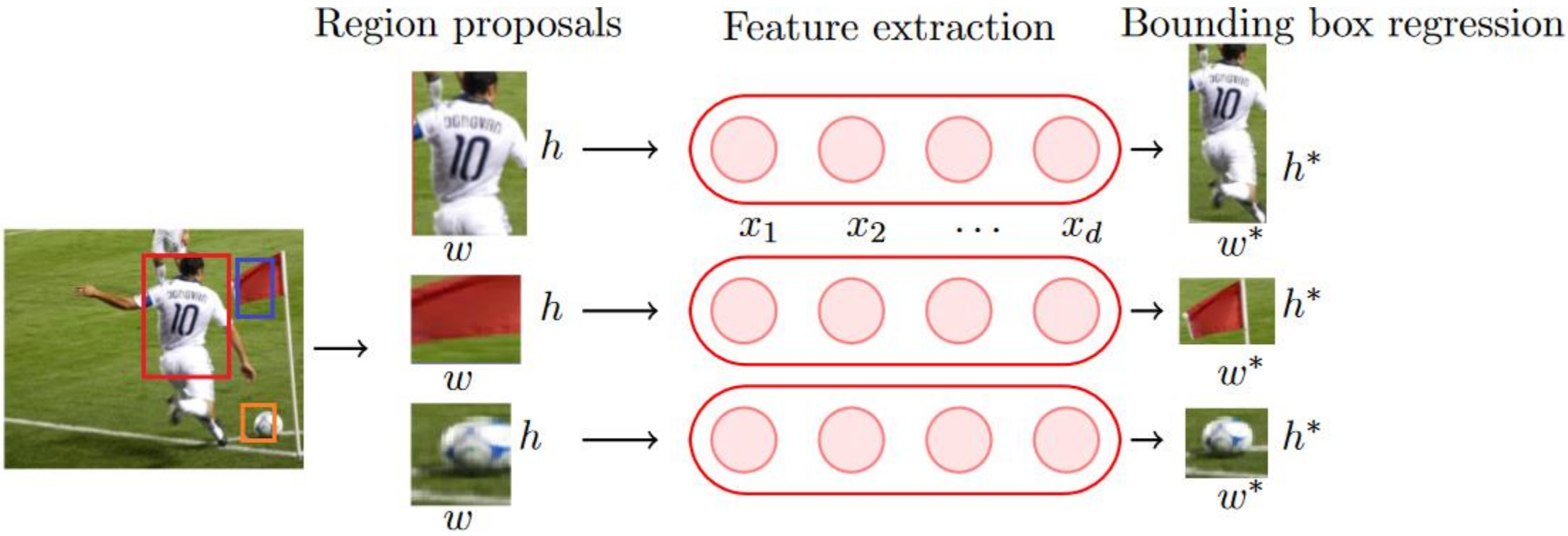
Reference systems

Slide credit : Ross Girshick

So, we do this

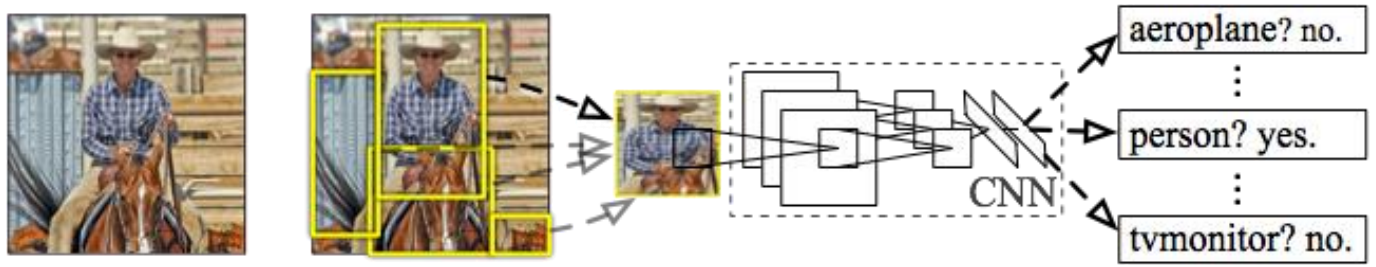

Region proposals
Feature extraction
Classifier
person
flag
ball
none
x_1
x_2
$\cdots$
x_d

A better approach



Classification + Regression

R-CNN: Regions with CNN features

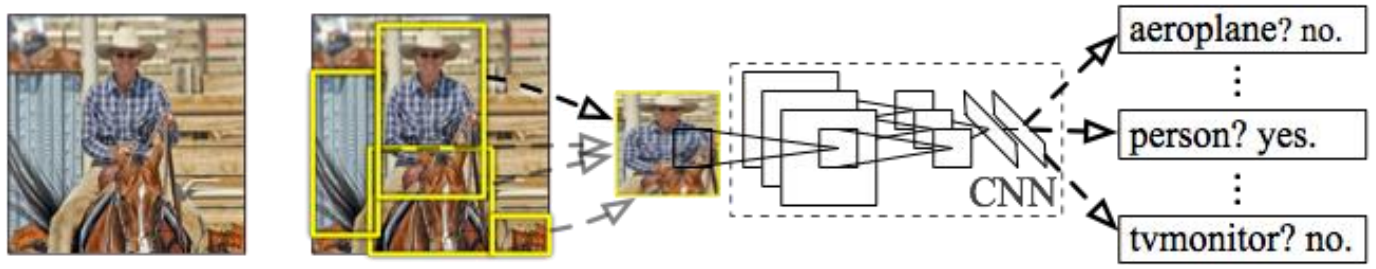


Input image | Extract region proposals (~2k / image) | Compute CNN features | Classify regions (linear SVM)

Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation
R. Girshick, J. Donahue, T. Darrell, J. Malik
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014

Slide credit : Ross Girshick

R-CNN at test time: Step 2



Input image

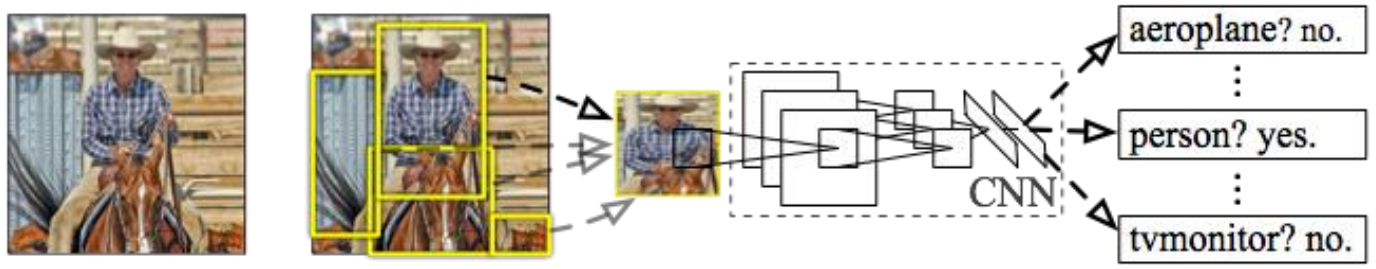
Extract region proposals (~2k / image)

Compute CNN features

a. Crop

Slide credit : Ross Girshick

R-CNN at test time: Step 2



Input image

Extract region proposals (~2k / image)

Compute CNN features

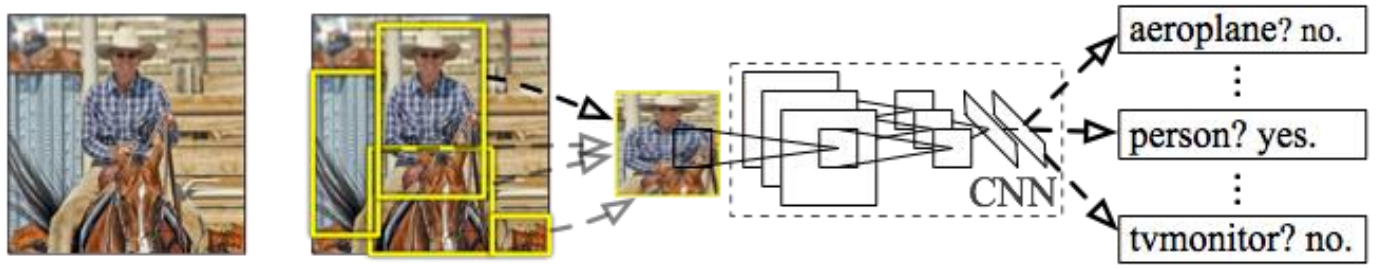
a. Crop

227 x 227

b. Scale (anisotropic)

Slide credit : Ross Girshick

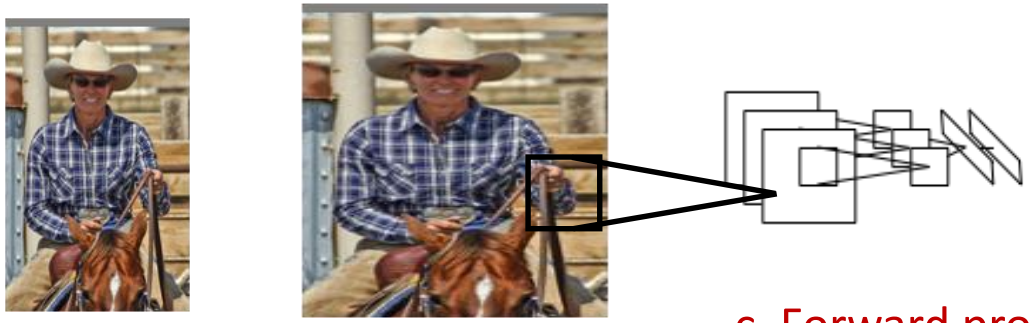
R-CNN at test time: Step 2



Input image

Extract region proposals (~2k / image)

Compute CNN features



1. Crop

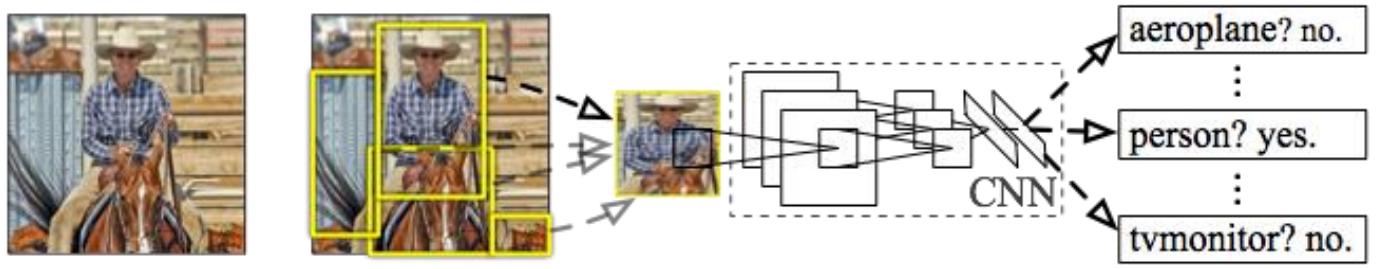
b. Scale (anisotropic)

c. Forward propagate

Output: "fc_7" features

Slide credit : Ross Girshick

R-CNN at test time: Step 3

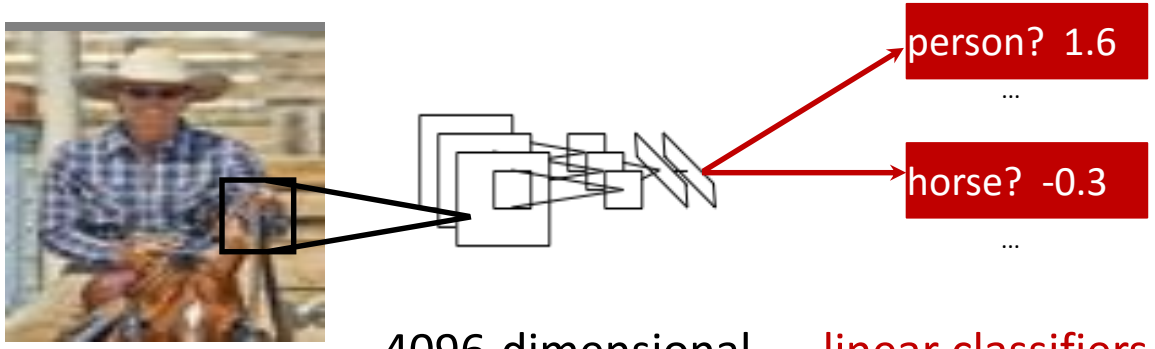


Input image

Extract region proposals (~2k / image)

Compute CNN features

Classify regions



Warped proposal

4096-dimensional fc_7 feature vector

linear classifiers (SVM or softmax)

Slide credit : Ross Girshick

Step 4: Object proposal refinement

Linear regression

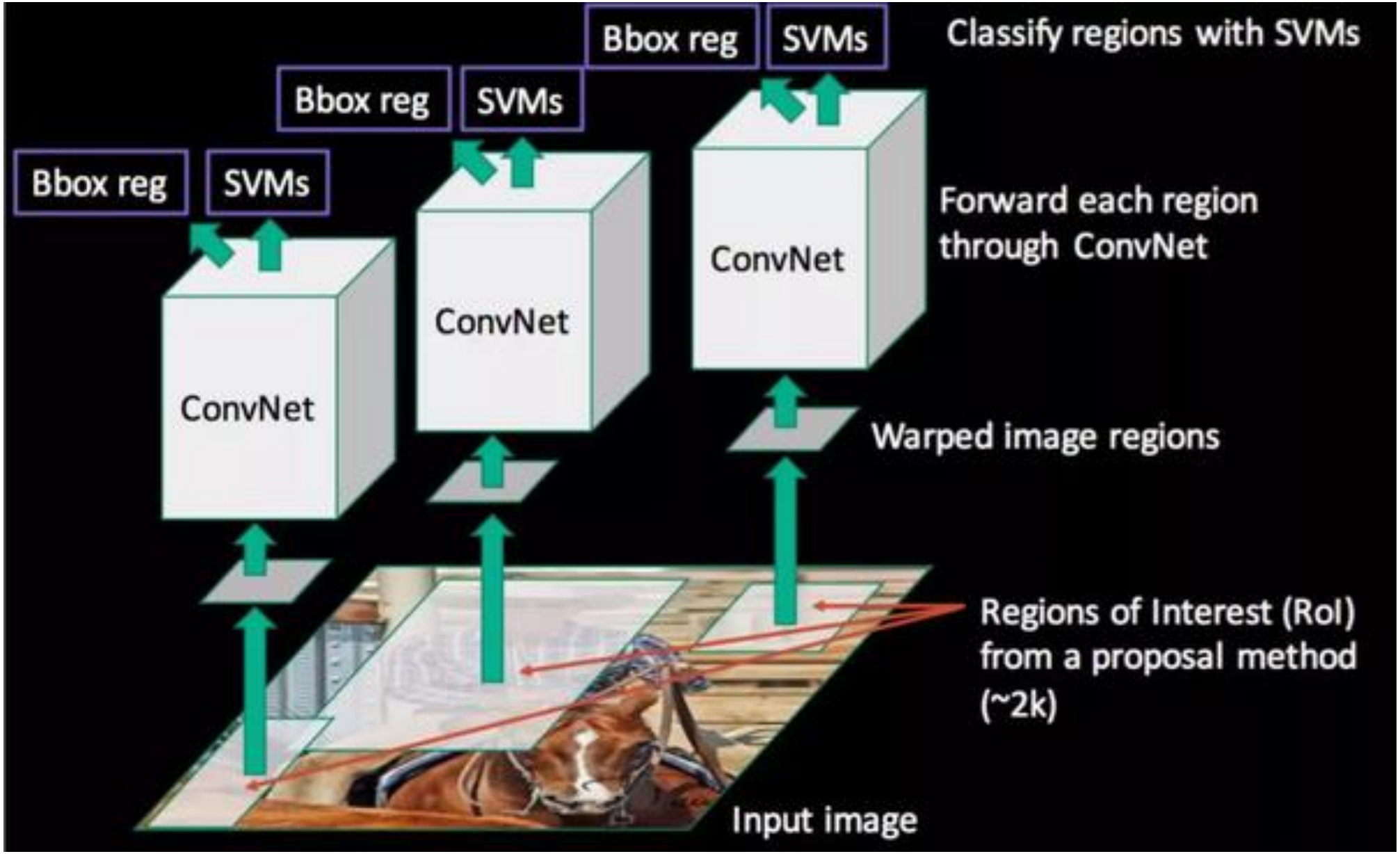
on CNN features

Original proposal

Predicted object bounding box

Bounding-box regression

Slide credit : Ross Girshick


Bbox reg
SVMs
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image

R-CNN results on PASCAL

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%

Reference systems

Slide credit : Ross Girshick

R-CNN results on PASCAL

	VOC 2007	VOC 2010
DPM v5 (Girshick et al. 2011)	33.7%	29.6%
UVA sel. search (Uijlings et al. 2013)		35.1%
Regionlets (Wang et al. 2013)	41.7%	39.7%
SegDPM (Fidler et al. 2013)		40.4%
R-CNN	54.2%	50.2%
R-CNN + bbox regression	58.5%	53.7%

Slide credit : Ross Girshick

Training R-CNN

- Train convolutional network on ImageNet classification
- *Finetune* on detection
  - Classification problem!
  - Proposals with IoU > 50% are positives
  - Sample fixed proportion of positives in each batch because of imbalance

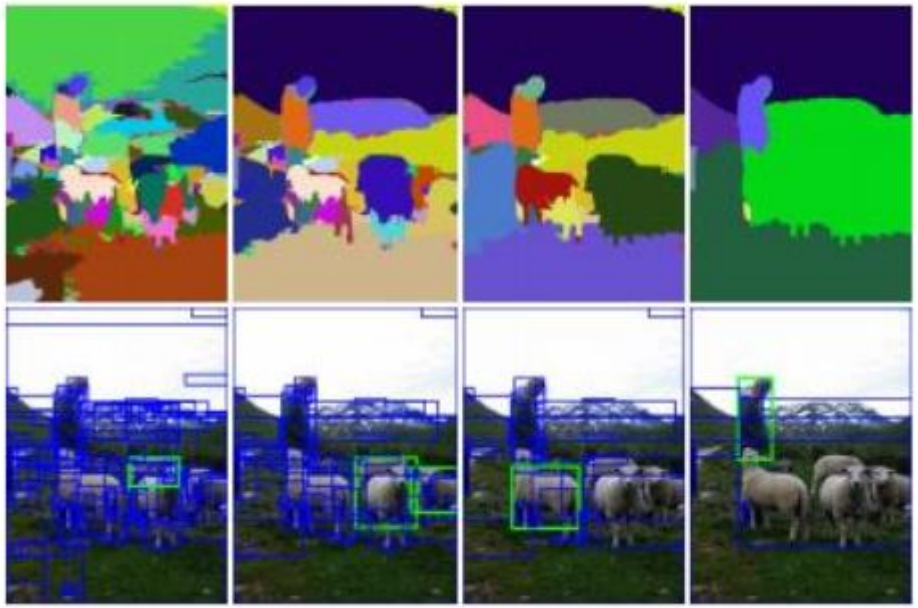
Other details - Non-max suppression

How do we deal with multiple detections on the same object?

Other details - Non-max suppression

- Go down the list of detections starting from highest scoring
- Eliminate any detection that overlaps highly with a higher scoring detection
- Separate, heuristic step

Selective search



Selective Search for region proposals

Does hierarchical clustering at different scales

For example the figures from left to right show clusters of increasing sizes

Such a hierarchical clustering is important as we may find different objects at different scales

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image

Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]

Initialise similarity set $S = \emptyset$

foreach *Neighbouring region pair* (r_i, r_j) **do**

    Calculate similarity $s(r_i, r_j)$

    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**

    Get highest similarity $s(r_i, r_j) = \max(S)$

    Merge corresponding regions $r_t = r_i \cup r_j$

    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$

    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$

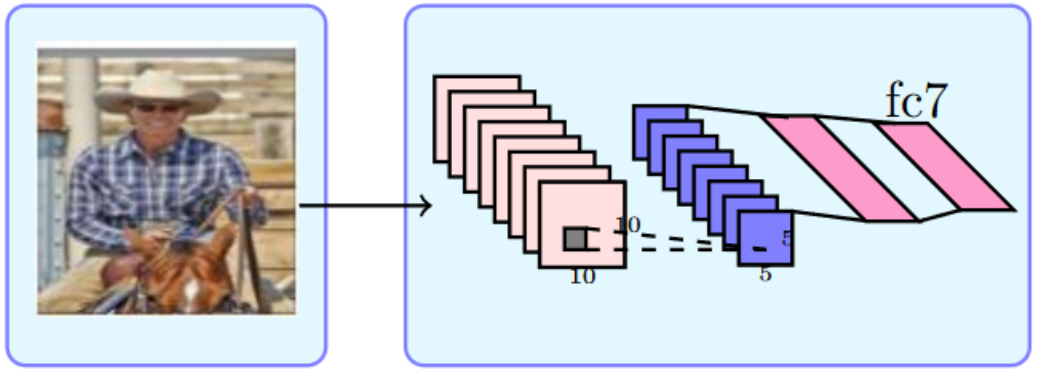
    Calculate similarity set S_t between r_t and its neighbours

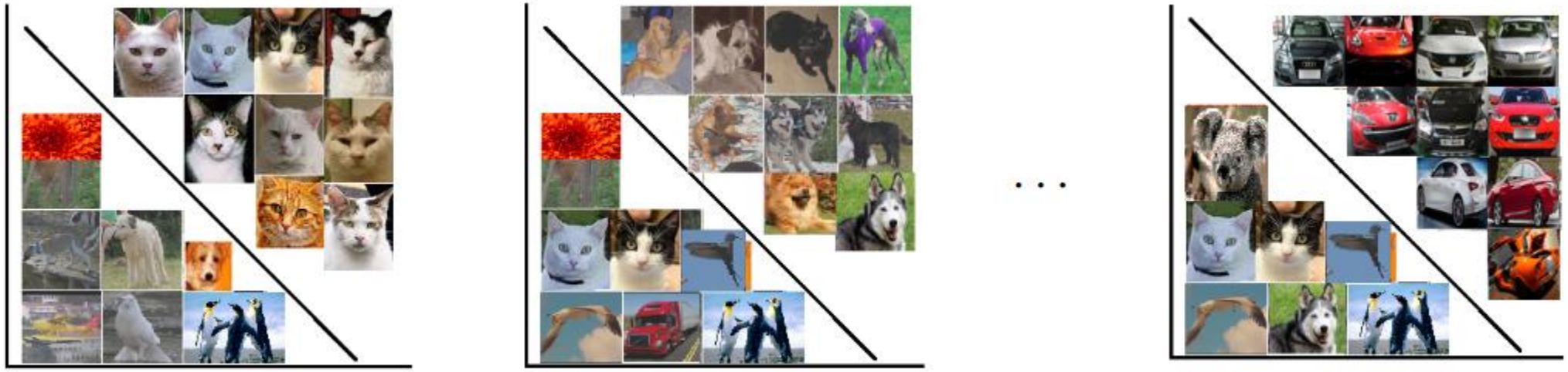
    $S = S \cup S_t$

    $R = R \cup r_t$

Extract object location boxes L from all regions in R

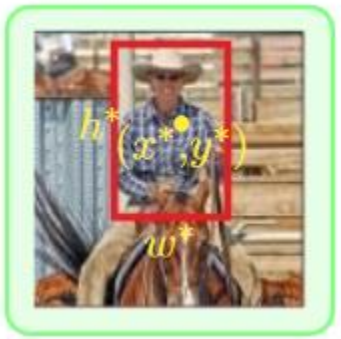
Fine-tune the CNN





Bounding box regressor

Proposed Box



True Box

The proposed regions may not be perfect

We want to learn four regression models which will learn to predict x^*, y^*, w^*, h^*

The input to our training algorithm is a set of N training pairs $\{(P^i, G^i)\}_{i=1,\ldots,N}$, where $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$ specifies the pixel coordinates of the center of proposal P^i's bounding box together with P^i's width and height in pixels. Hence forth, we drop the superscript i unless it is needed. Each ground-truth bounding box G is specified in the same way: $G = (G_x, G_y, G_w, G_h)$. Our goal is to learn a transformation that maps a proposed box P to a ground-truth box G.

Bounding box regressor

Learnable parameter

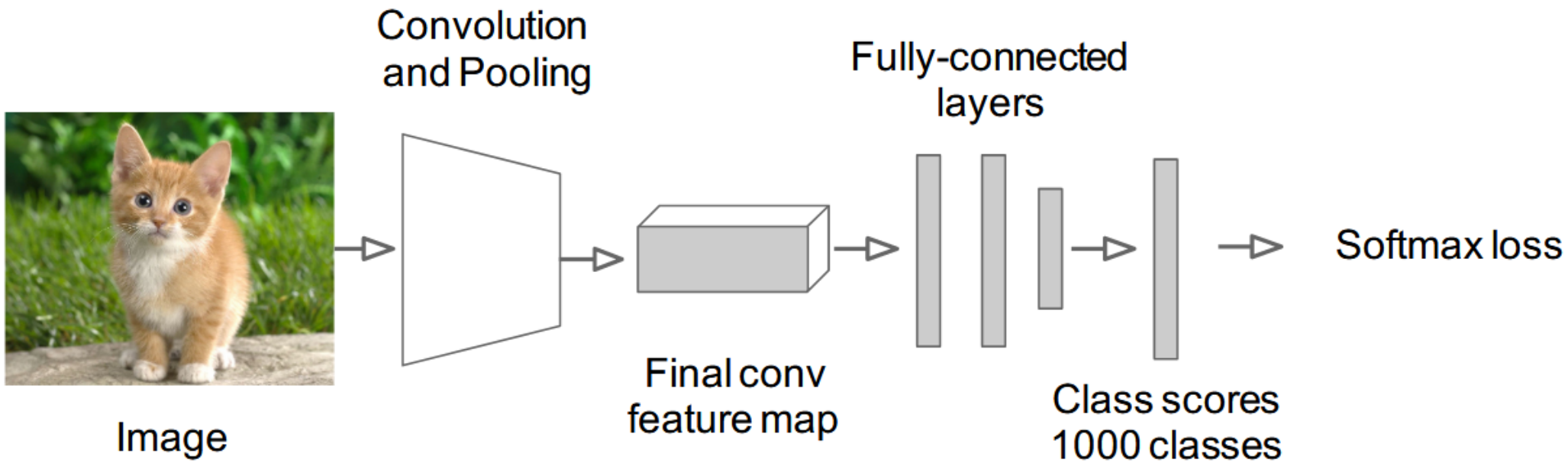
$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2$$

Normalized difference between predicted and true box

$$\begin{aligned} t_x &= (G_x - P_x)/P_w \\ t_y &= (G_y - P_y)/P_h \\ t_w &= \log(G_w/P_w) \\ t_h &= \log(G_h/P_h). \end{aligned}$$

$$\begin{aligned} \hat{G}_x &= P_w d_x(P) + P_x \\ \hat{G}_y &= P_h d_y(P) + P_y \\ \hat{G}_w &= P_w \exp(d_w(P)) \\ \hat{G}_h &= P_h \exp(d_h(P)). \end{aligned}$$

Step 1: Train (or download) a classification model for ImageNet (AlexNet)



Step 2: Fine-tune model for detection

- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images

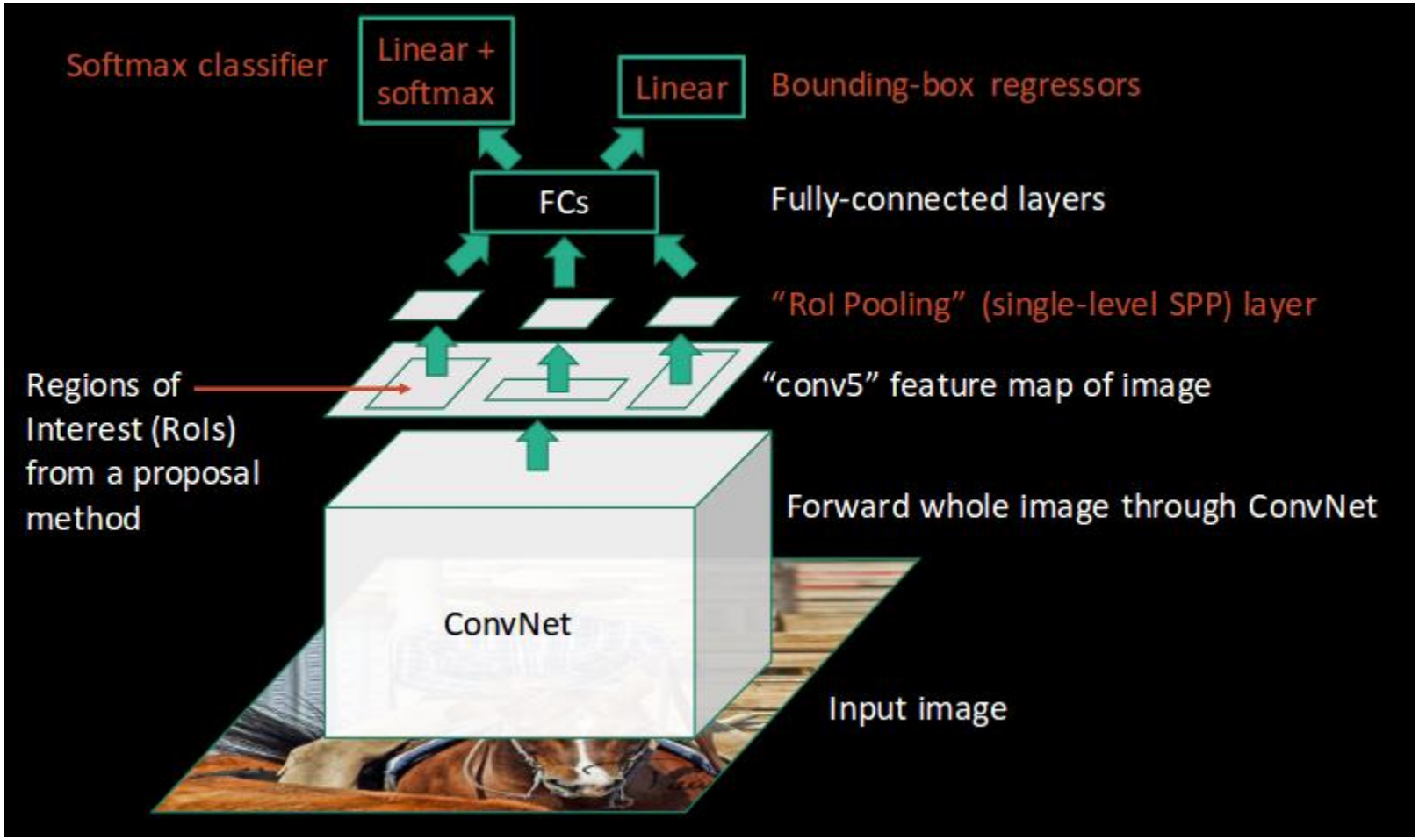
Step 3: Extract features

- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!

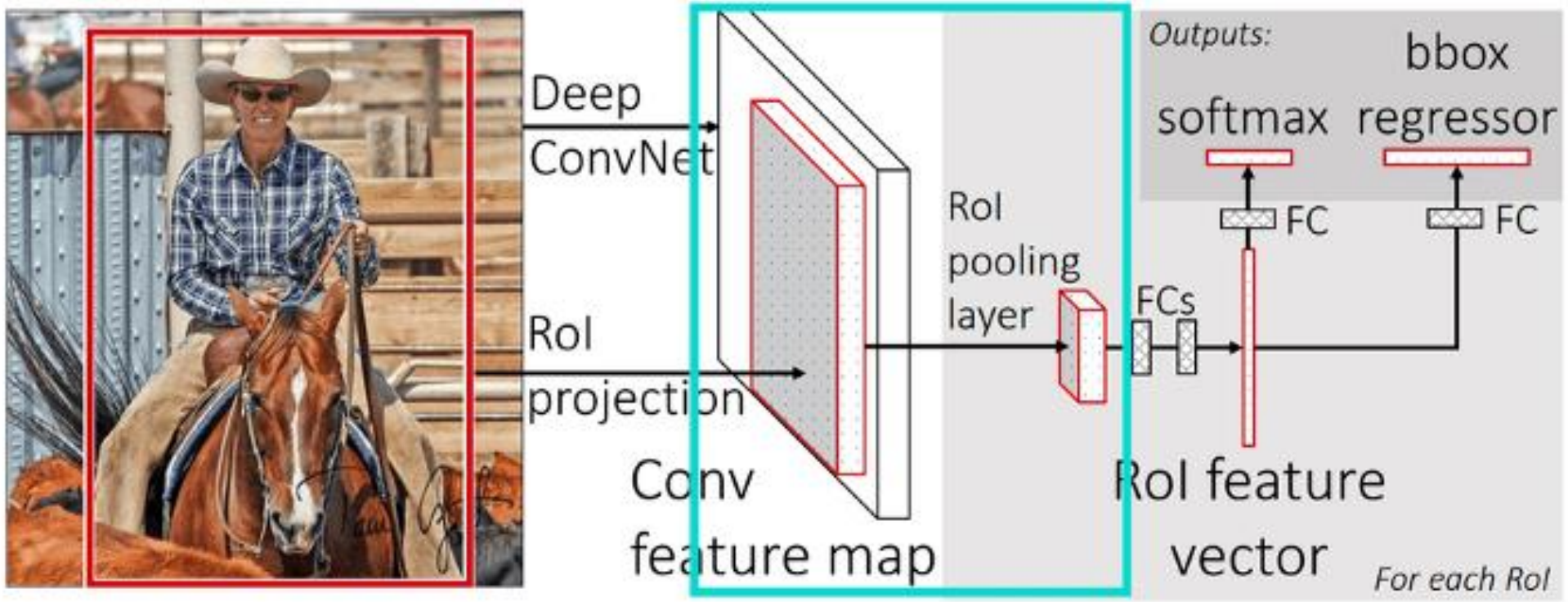
Step 4: Train one binary SVM per class to classify region features

Step 5 (bbox regression): For each class, train a linear regression model to map from cached features to offsets to GT boxes to make up for "slightly wrong" proposals

Fast r-CNN


Softmax classifier
Linear + softmax
Linear
Bounding-box regressors
FCs
Fully-connected layers
"RoI Pooling" (single-level SPP) layer
Regions of Interest (RoIs) from a proposal method
"conv5" feature map of image
Forward whole image through ConvNet
ConvNet
Input image

Fast r-CNN a closer look



Time comparison

	R-CNN	Fast R-CNN
Training Time:	84 hours	**9.5 hours**
(Speedup)	1x	**8.8x**
Test time per image	47 seconds	**0.32 seconds**
(Speedup)	1x	**146x**
mAP (VOC 2007)	66.0	**66.9**

Faster!

FASTER!

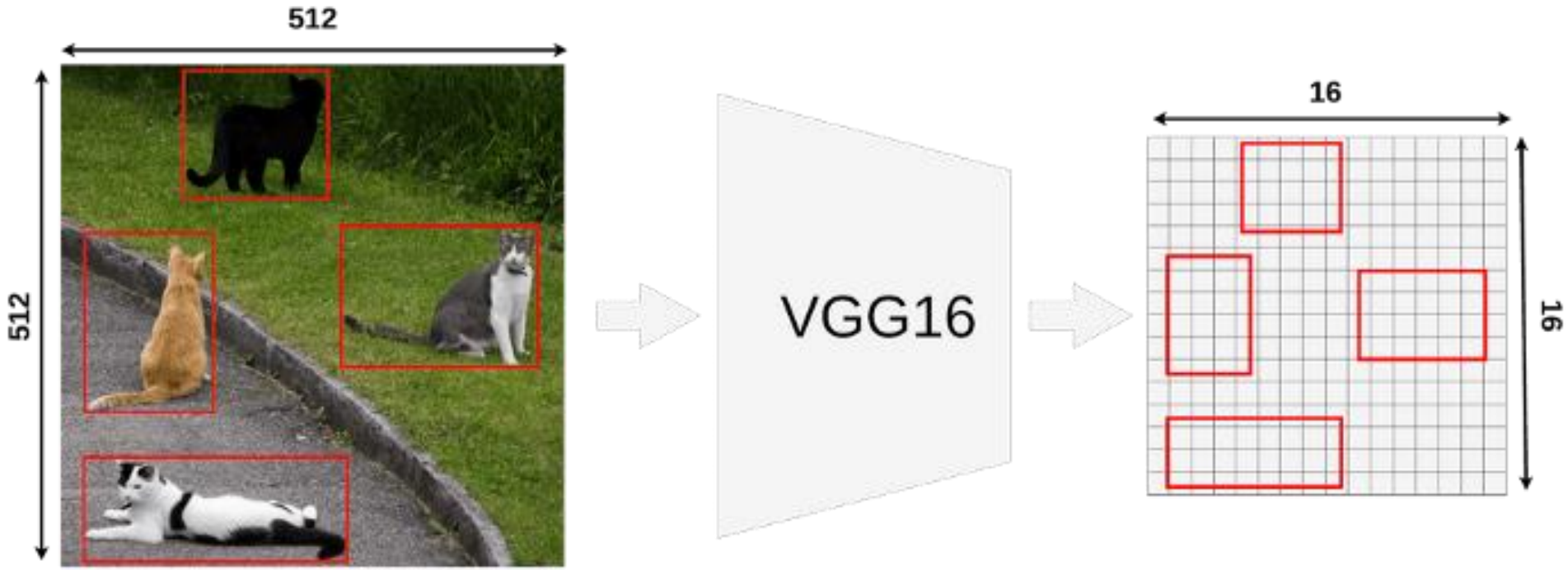
Better!

Using VGG-16 CNN on Pascal VOC 2007 dataset

Two issues

- How to find the location in the feature maps for a given roi
- How to re-shape the rois in the feature maps so they can be fed to the fc layers

Transform the original roi into feature maps



Problems

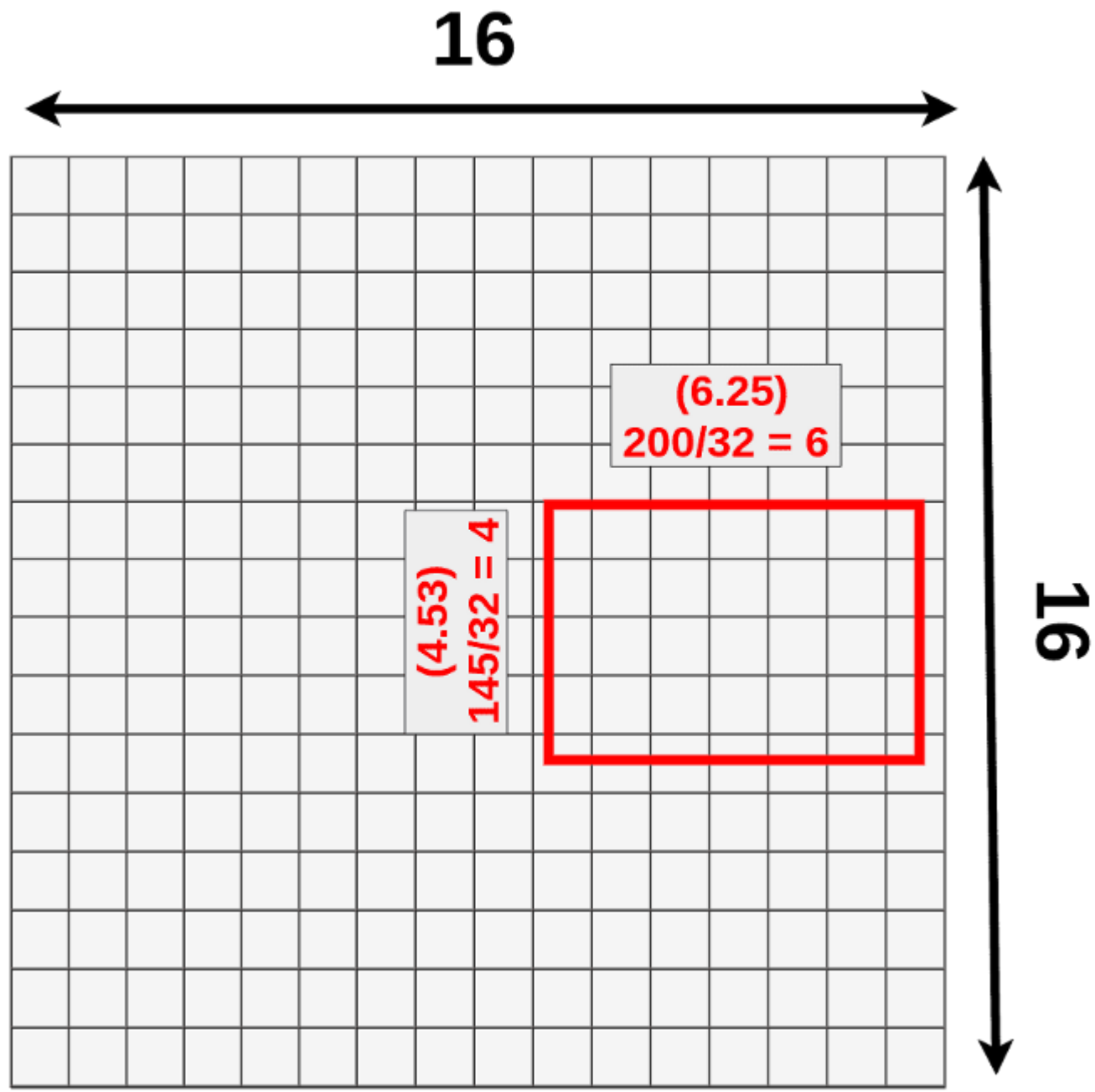
- The conversion may have quantization problem.
- Remember each box is represented by (x, y, w, h)
- Since the reduction is 1/16th the original image size in VGG, x/16, y/16 may be fractions.

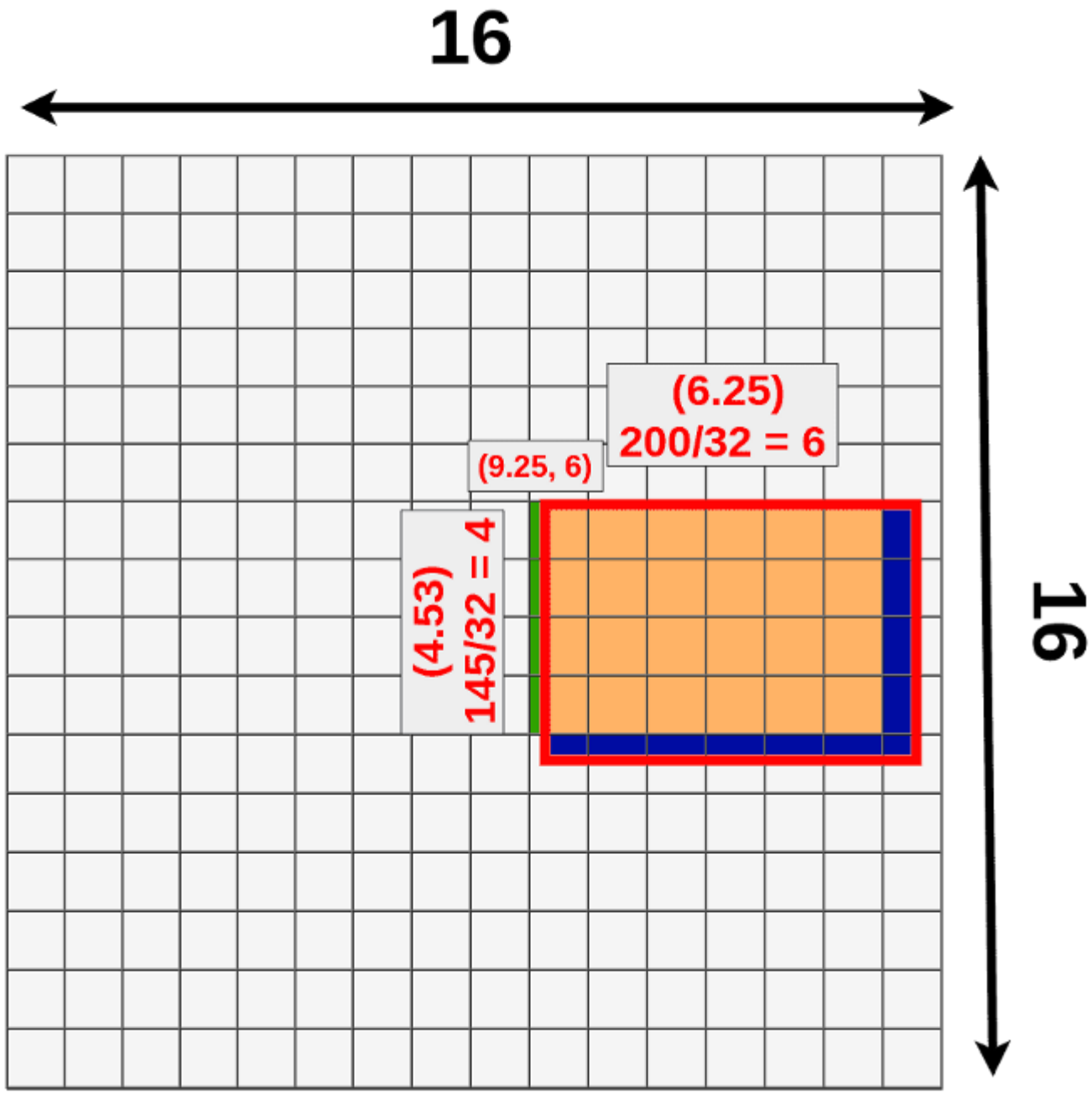
width: 200/32 = 6.25

height: 145/32 = ~4.53

x: 296/32 = 9.25

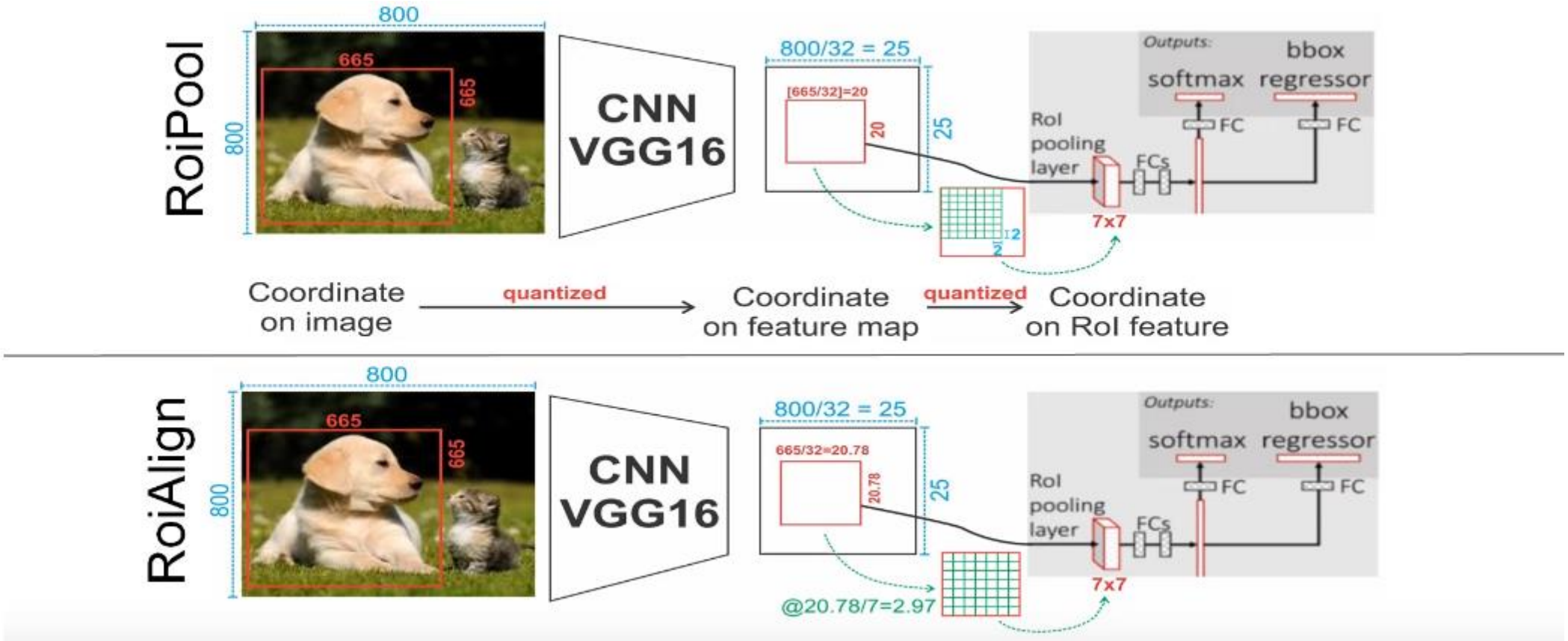
y: 192/32 = 6


16
16
(6.25)
200/32 = 6
(4.53)
145/32 = 4



Green – displacement
Blue – loss of information

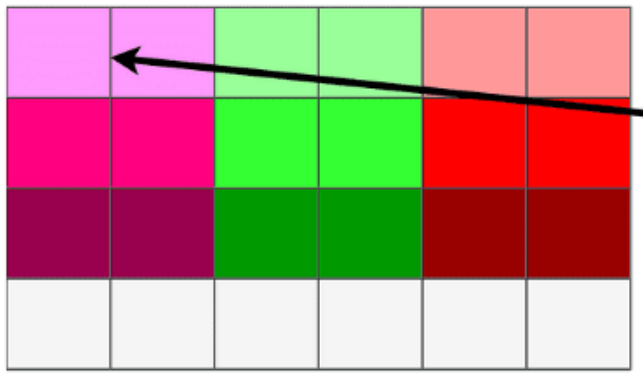
Roipool and Roialign



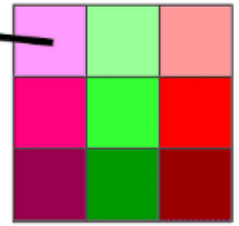
Roi-Pool

4x6 RoI

1x2 (4/3 = 1 x 6/3 = 2)



3x3 RoI Pooling

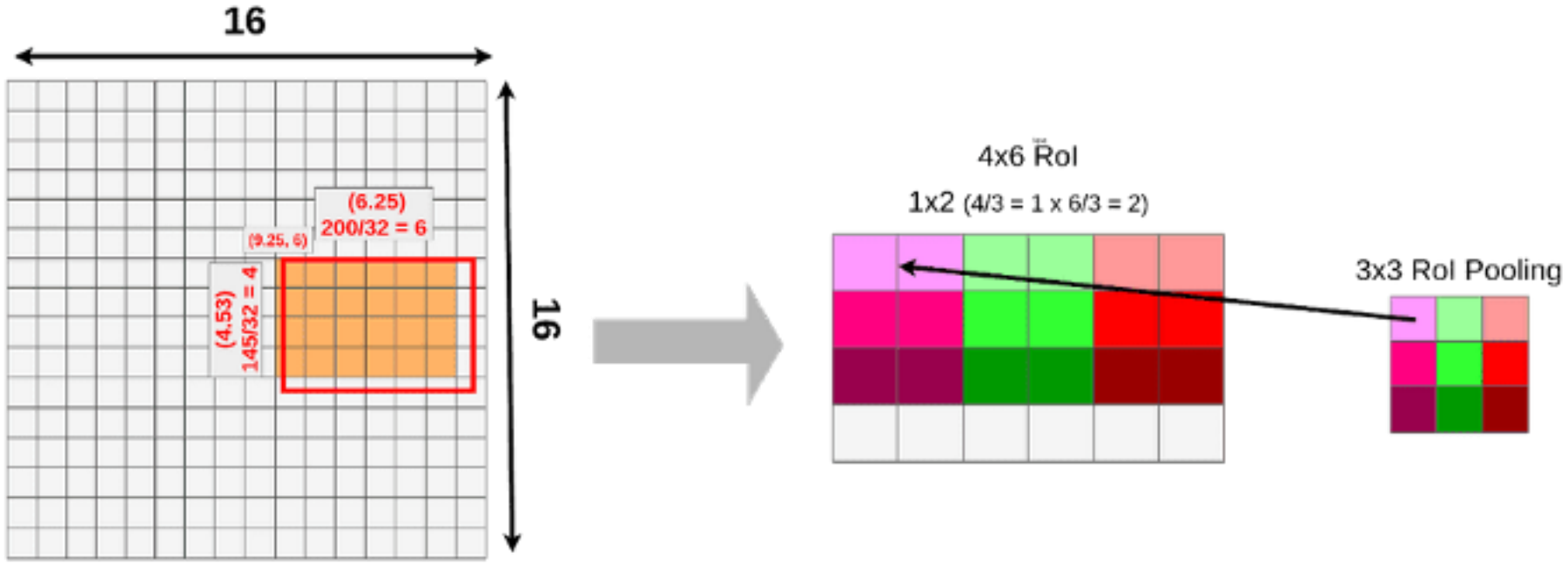


4x6 RoI

0.1	0.2	0.3	0.4	0.5	0.6
1	0.7	0.2	0.6	0.1	0.9
0.9	0.8	0.7	0.3	0.5	0.2

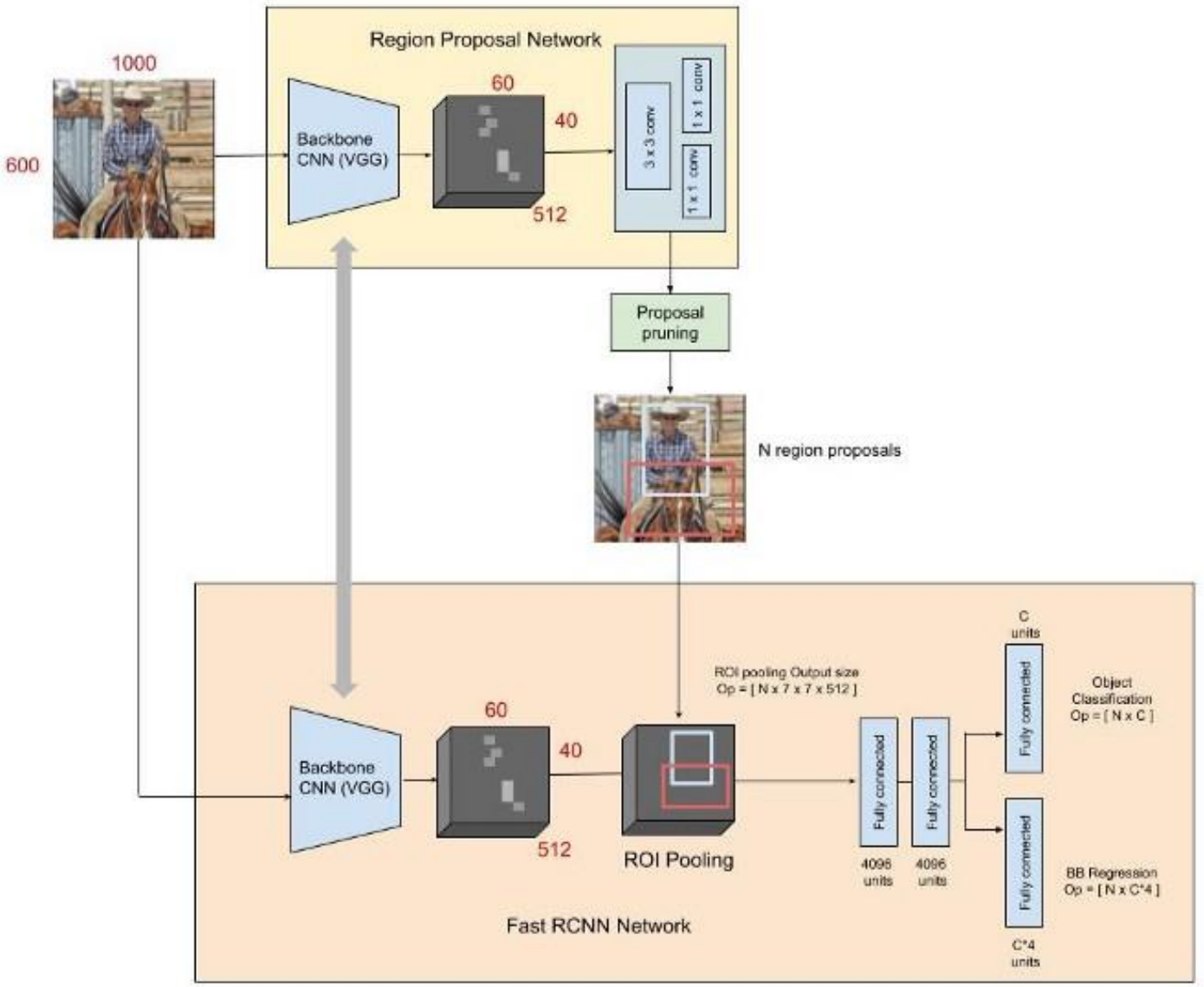
3x3 RoI Pooling

Quantization twice

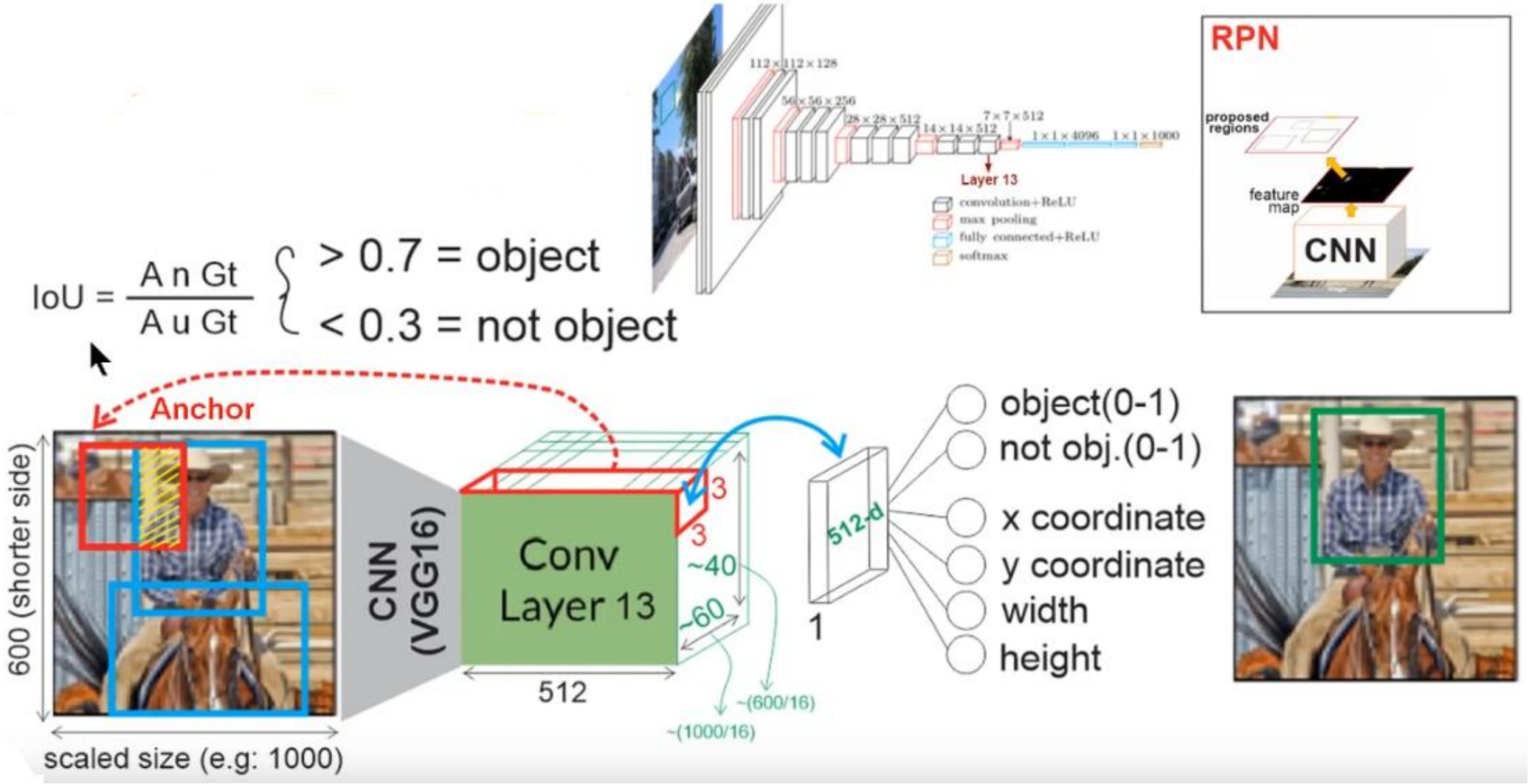


Faster r-CNN

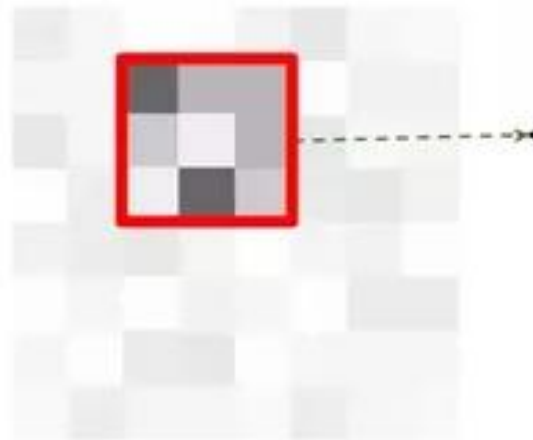
Can we get rid off the proposal generation by an ad-hoc technique?

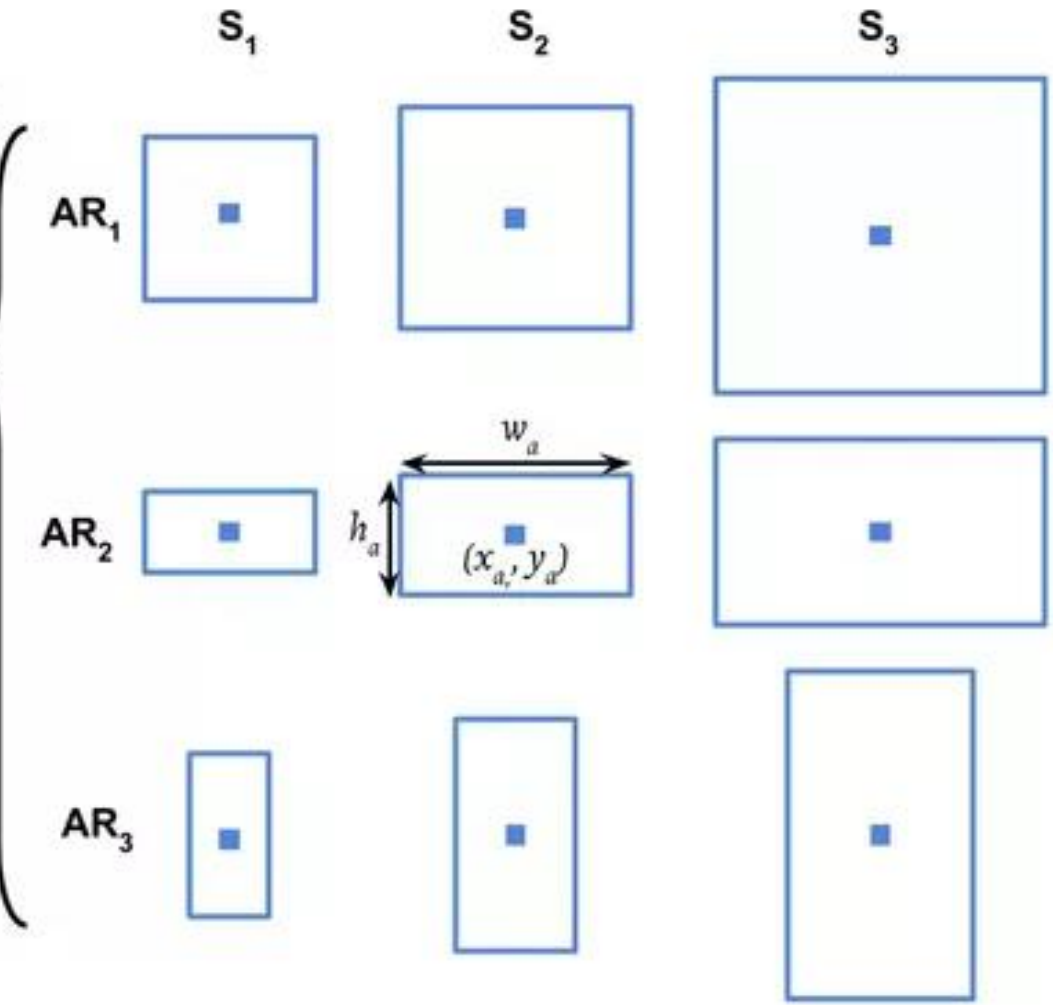


Region proposal network



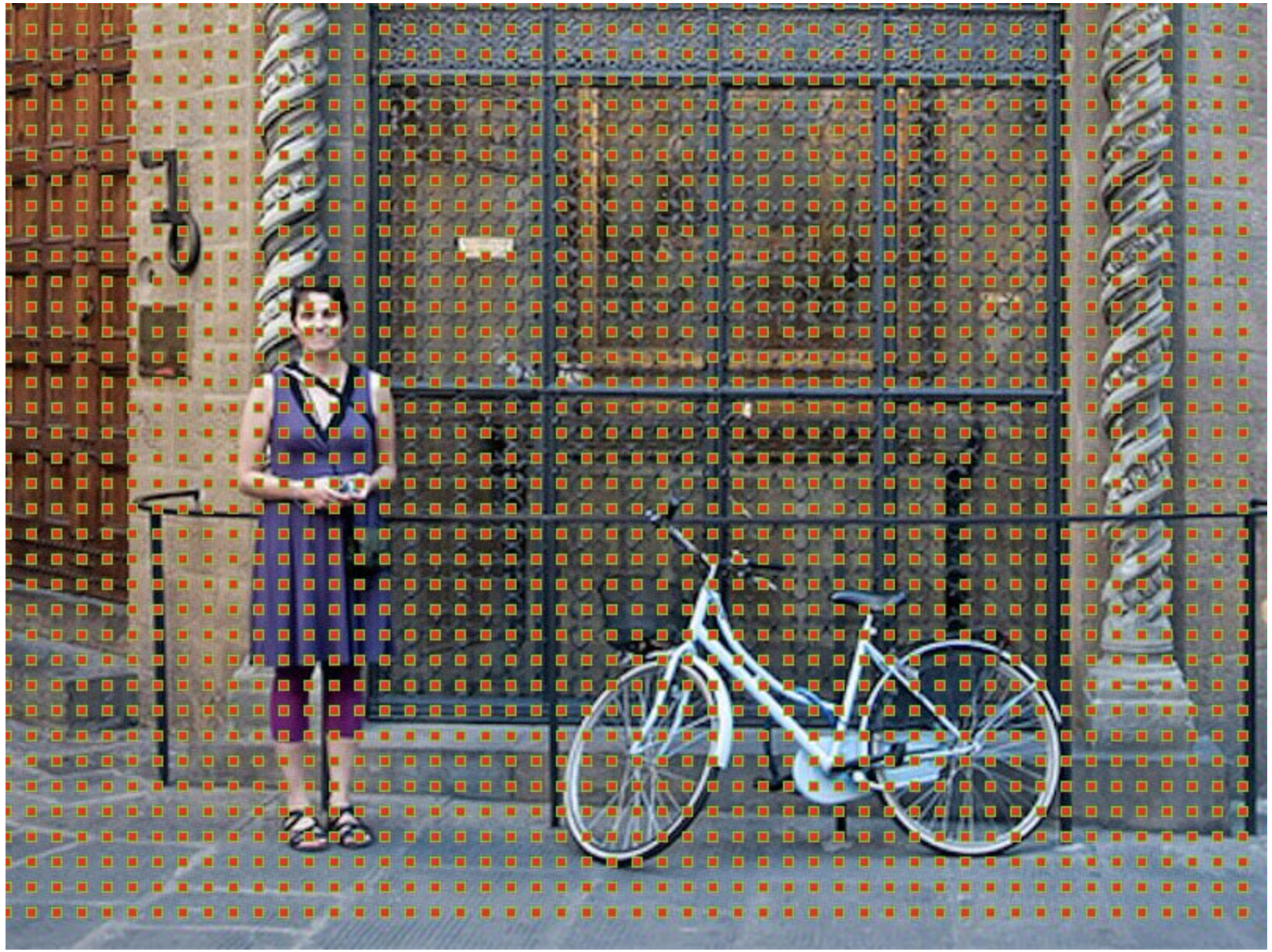
Generate 9 anchors for each sliding window on conv. feature map

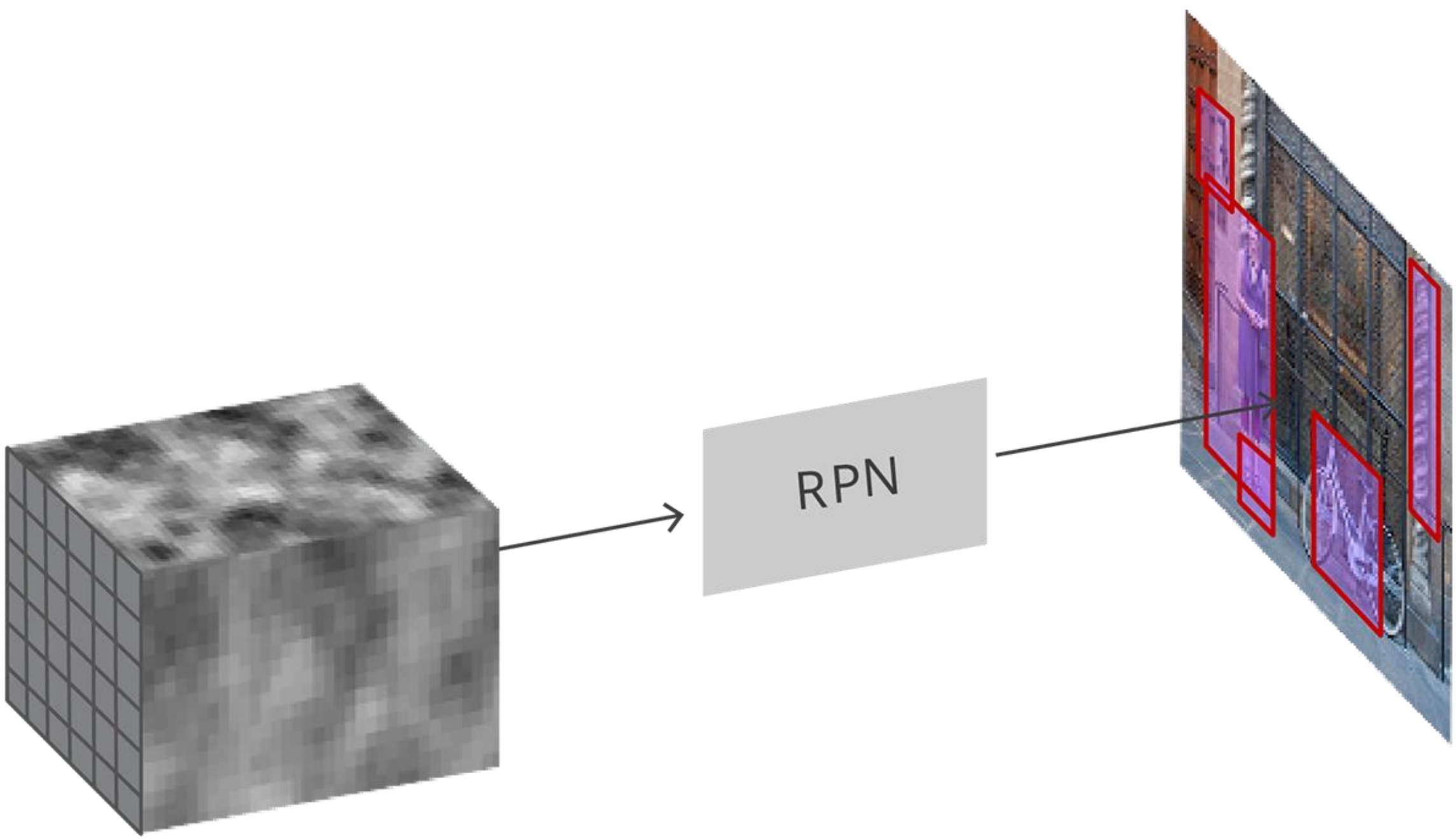




w_a: *anchor's width*
h_a: *anchor's height*
x_a, y_a: *anchor's center*

@vmirly




RPN

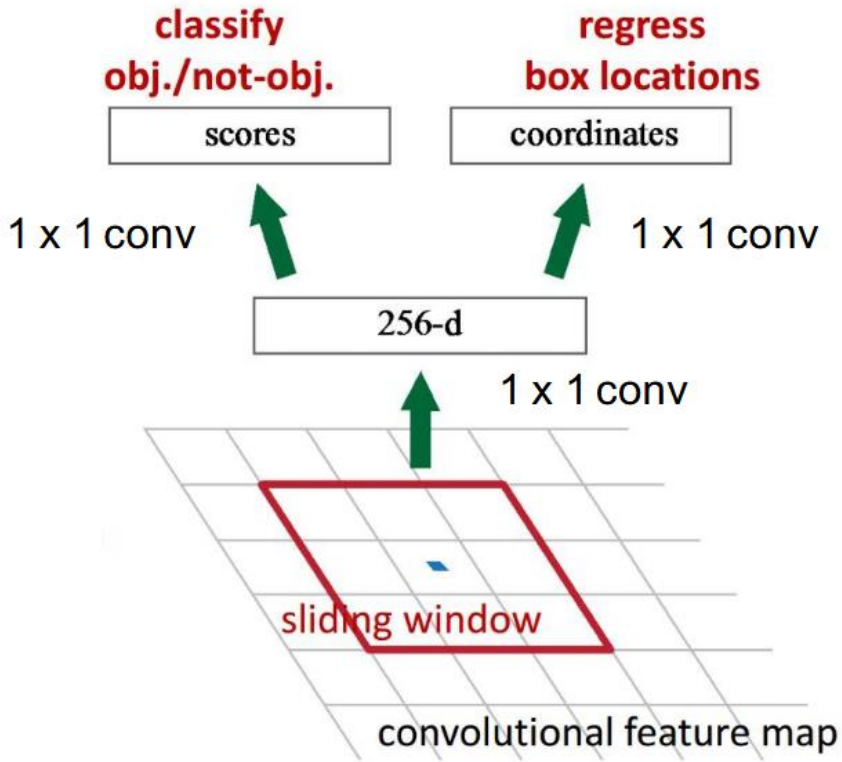
RPN

Slide a small window on the feature map

Build a small network for:
- classifying object or not-object, and
- regressing bbox locations

Position of the sliding window provides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window

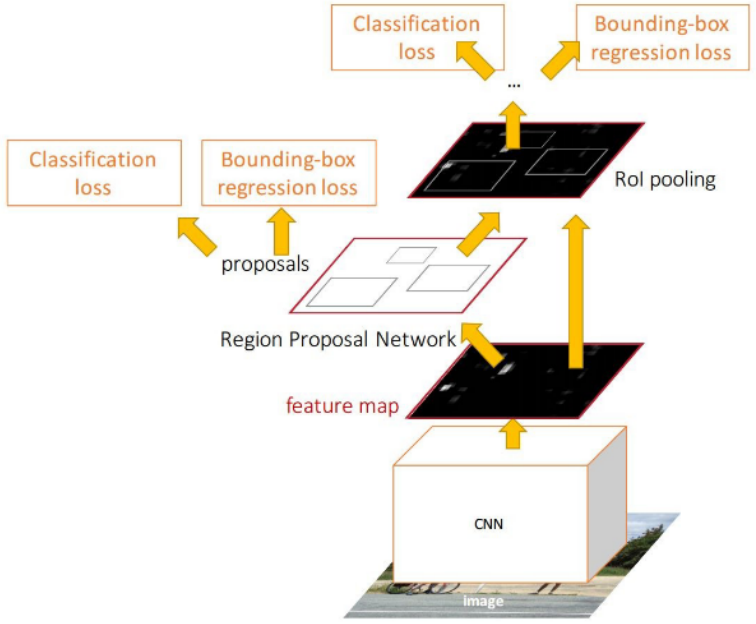


$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*)$$

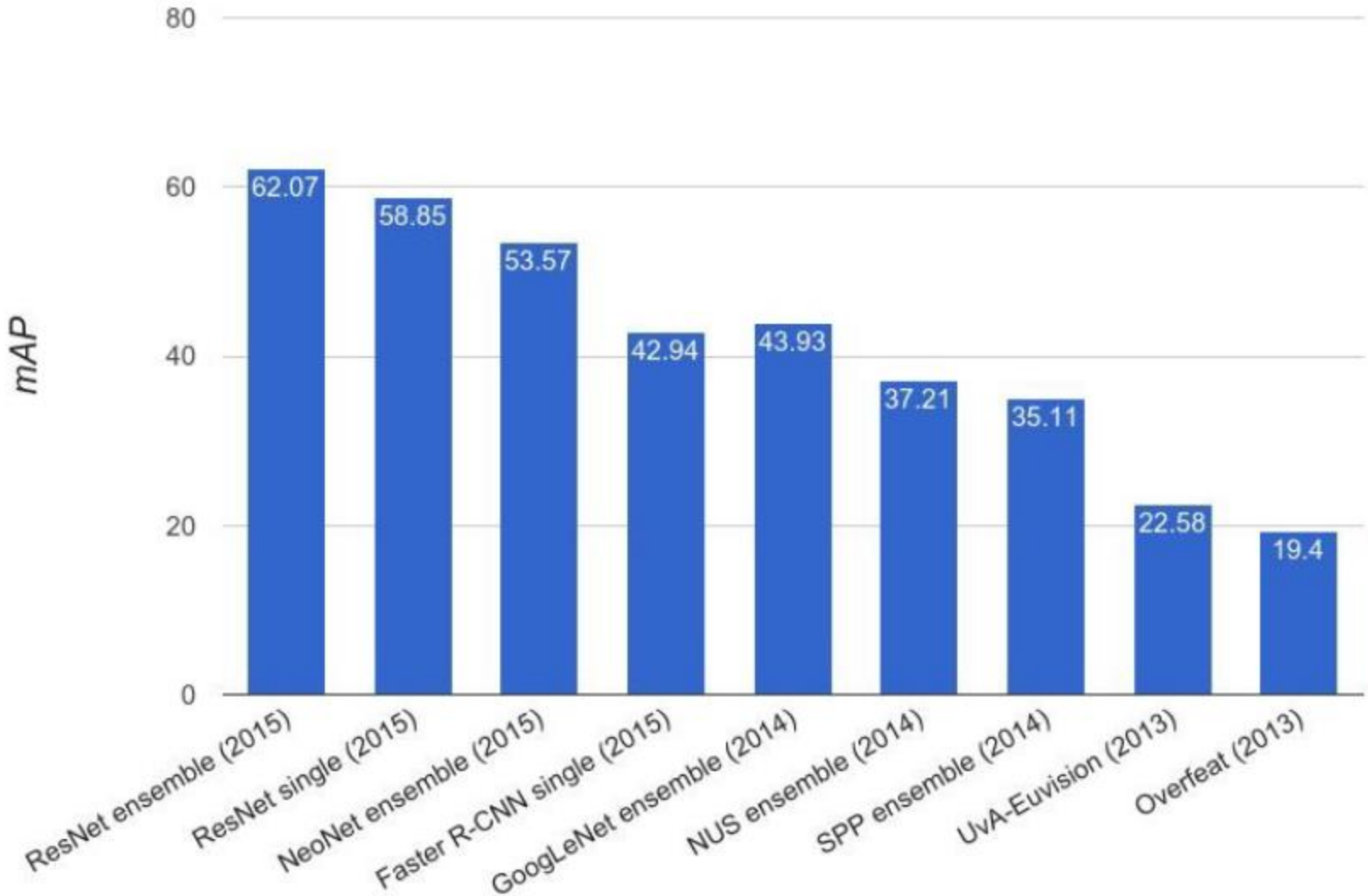
Faster rCNN training

In the paper: Ugly pipeline
- Use alternating optimization to train RPN, then Fast R-CNN with RPN proposals, etc.
- More complex than it has to be

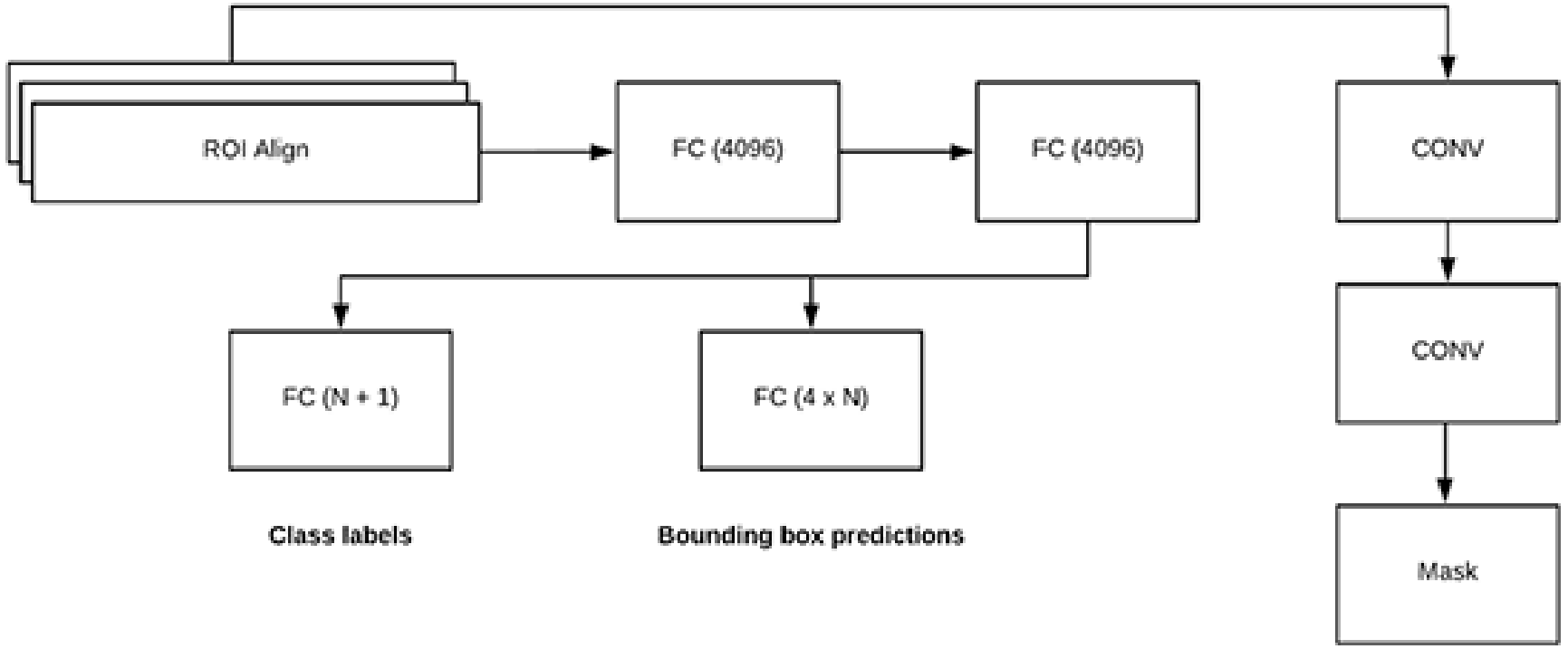
Since publication: Joint training!
One network, four losses
- RPN classification (anchor good / bad)
- RPN regression (anchor -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)



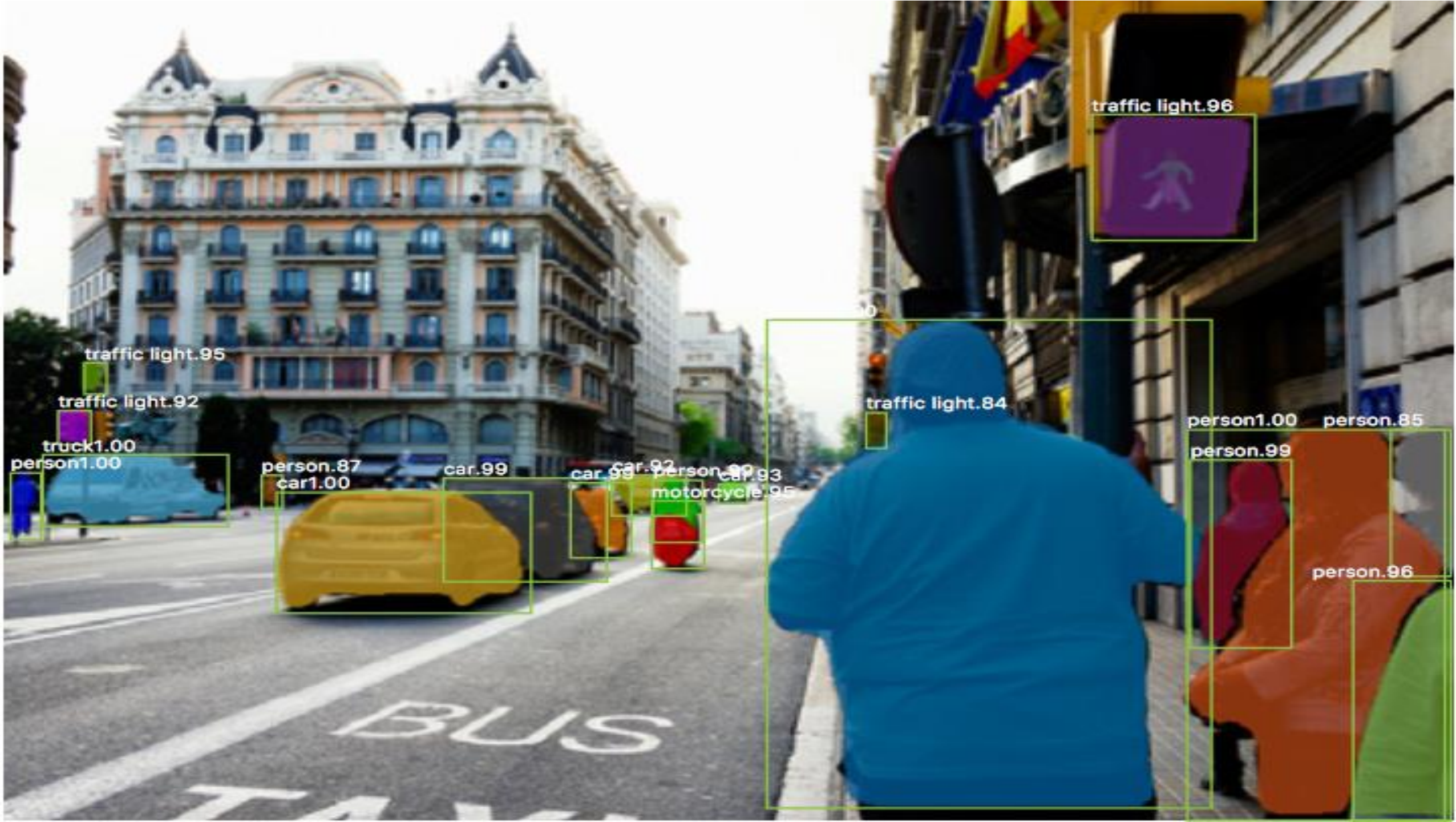
ImageNet Detection (mAP)



Mask r-CNN



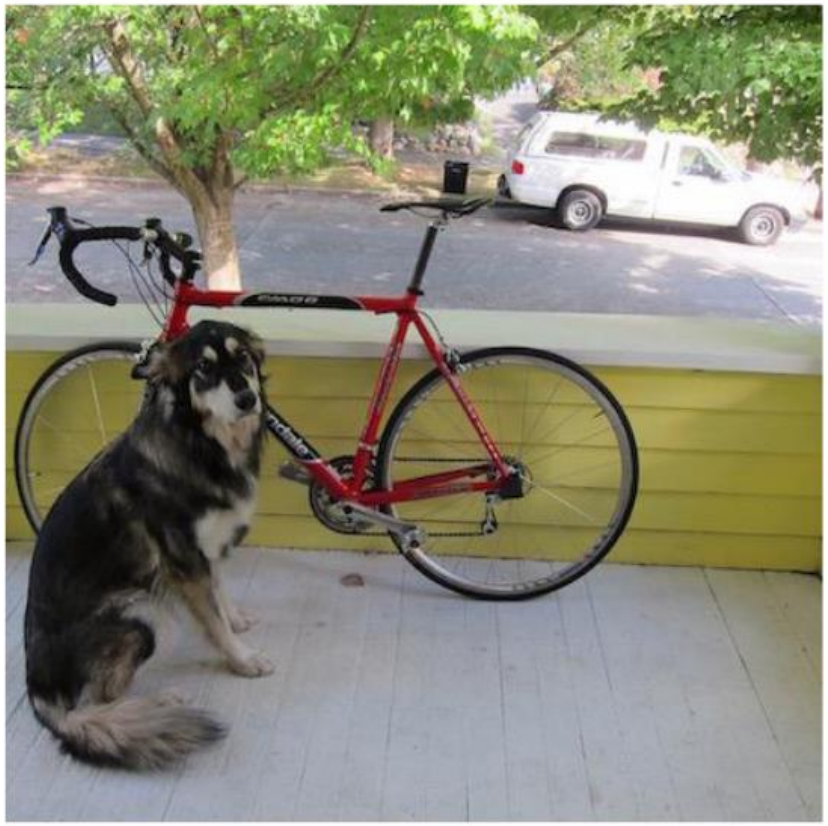
- ✓ Faster r-CNN detector + FCN segmentation
- ✓ Binary segmentation inside each bounding box

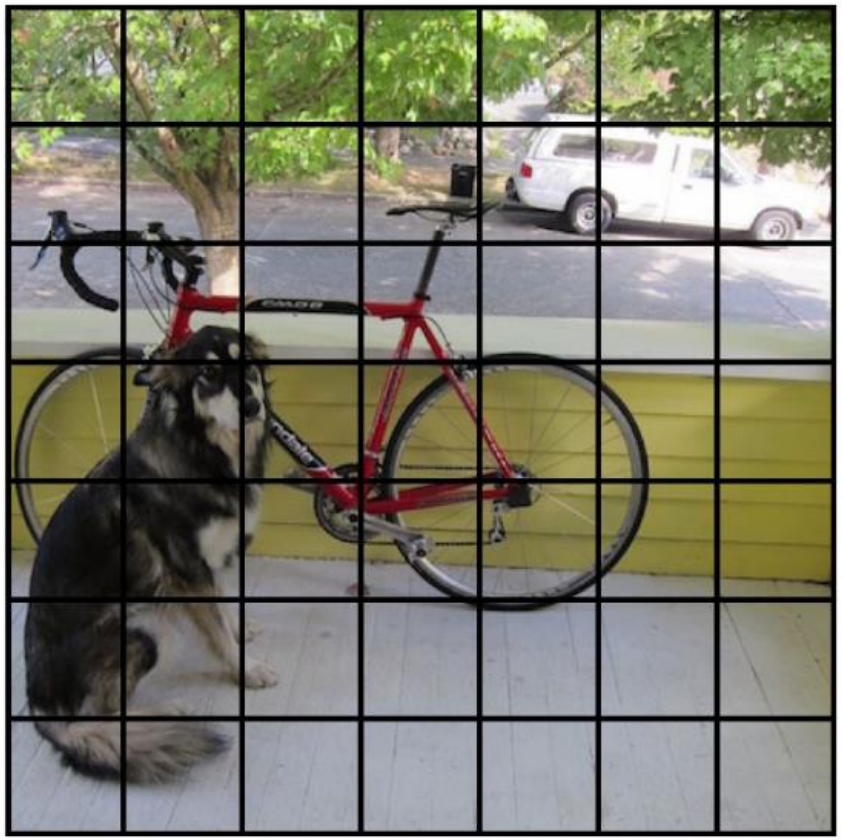

traffic light.96
traffic light.95
traffic light.92
truck1.00
person1.00
person.87
car1.00
car.99
car.99
car.92
motorcycle.95
car.93
traffic light.84
person1.00
person.85
person.99
person.96
BUS

Yolo

- Extremely fast (45 frames per second)
- Reason globally on the entire image
- Can learn generalizable representations

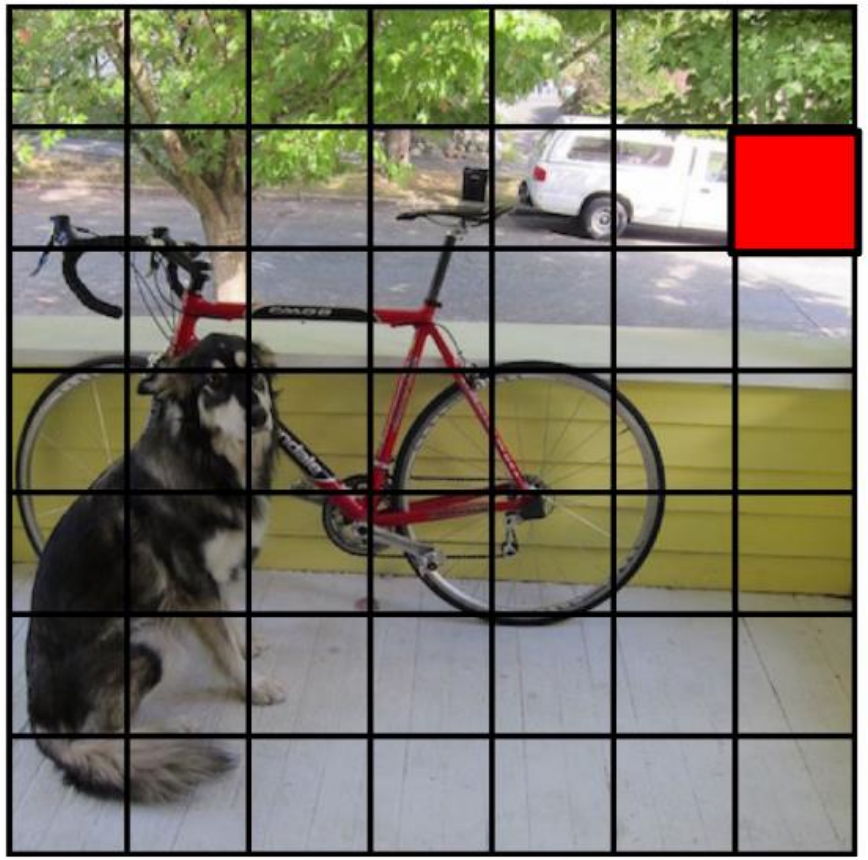
Stages

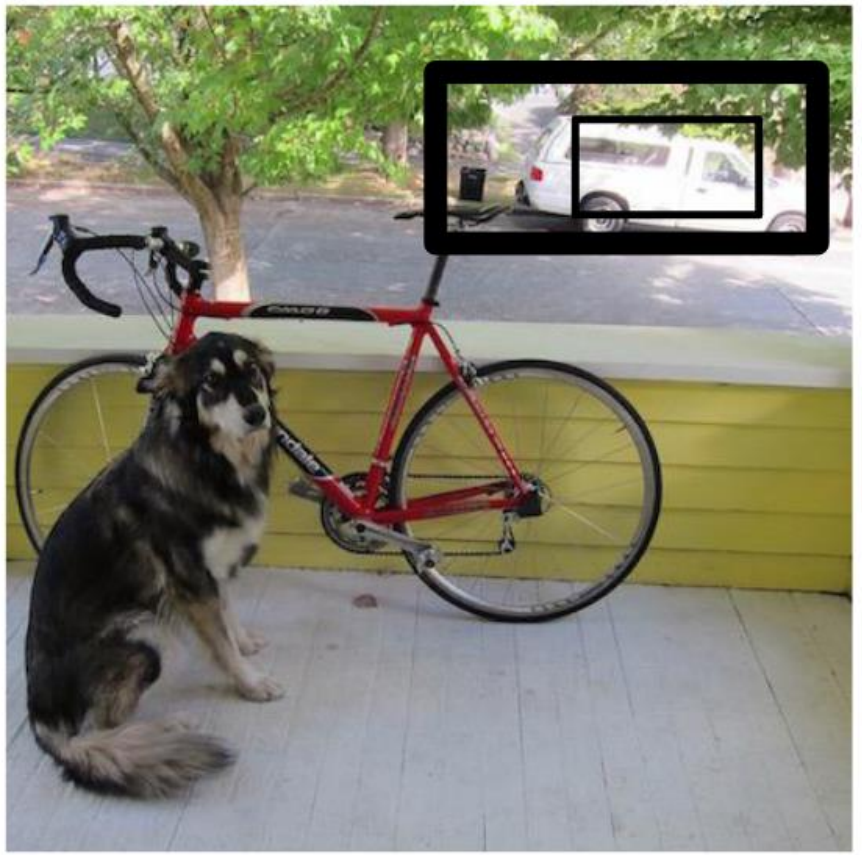




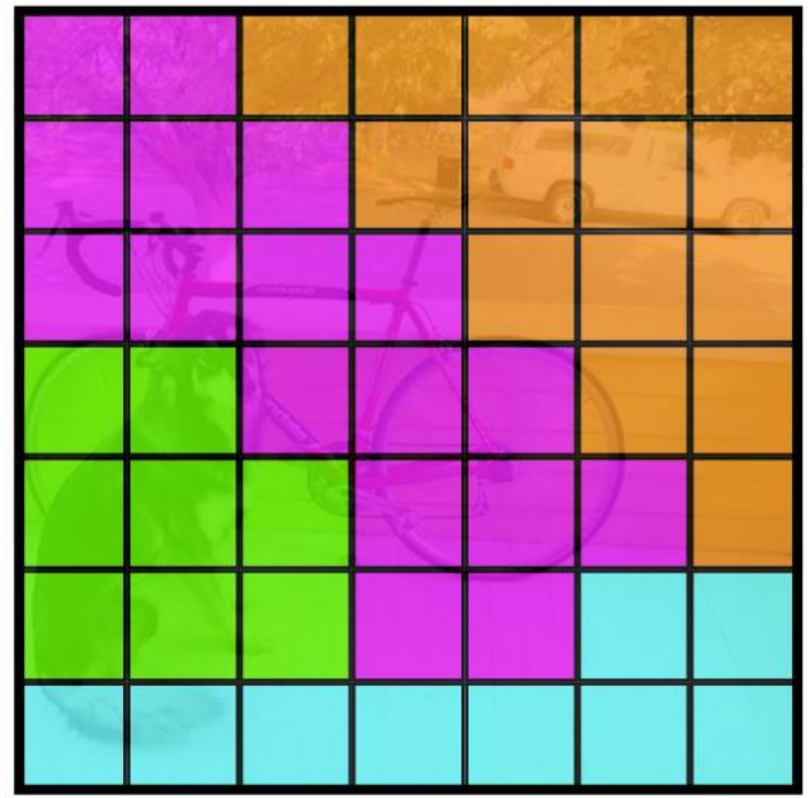
7*7 grid

Each cell predicts B=2 boxes(x,y,w,h) and confidences of each box: P(Object)



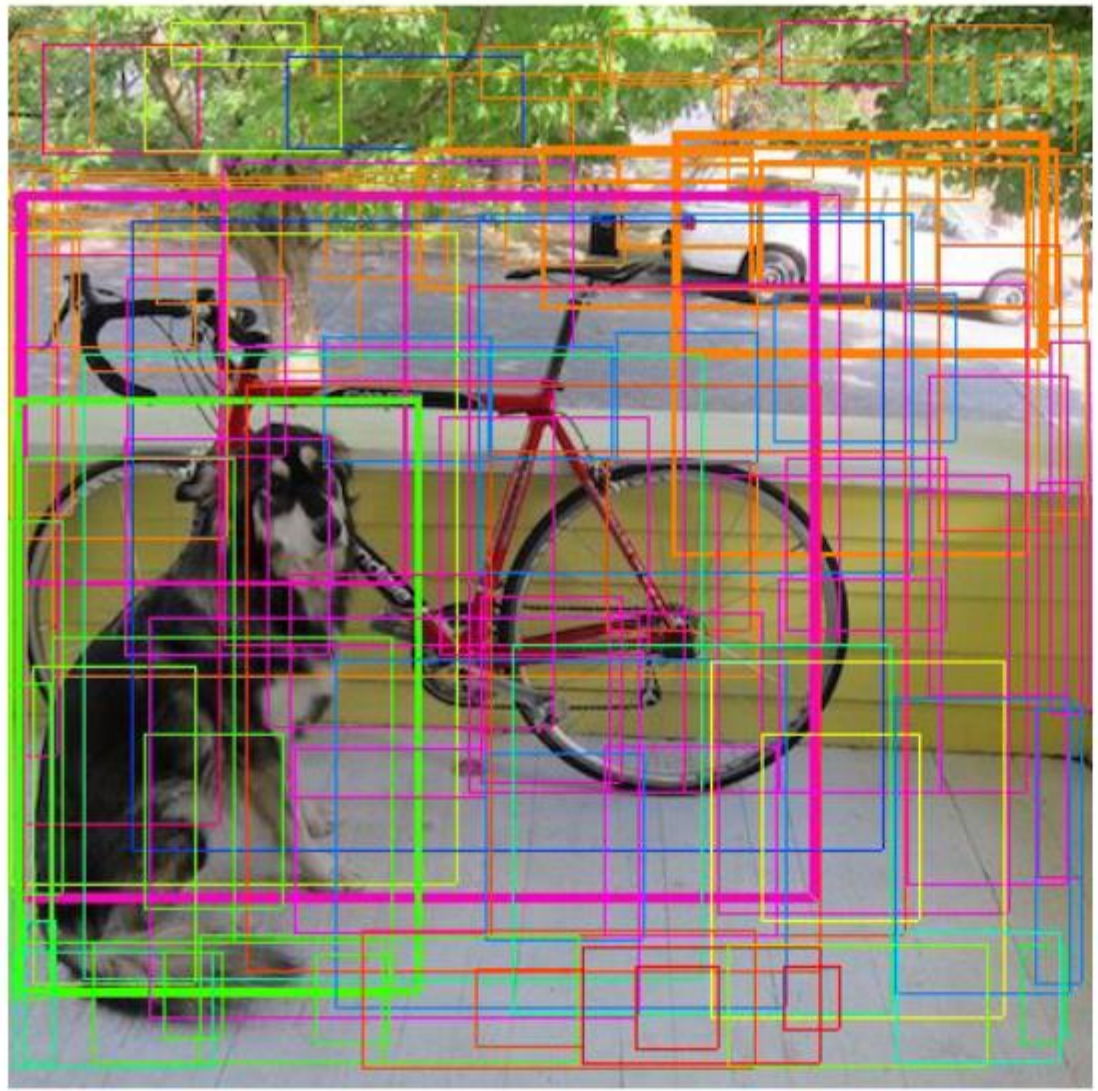


Each cell also predicts a class probability.



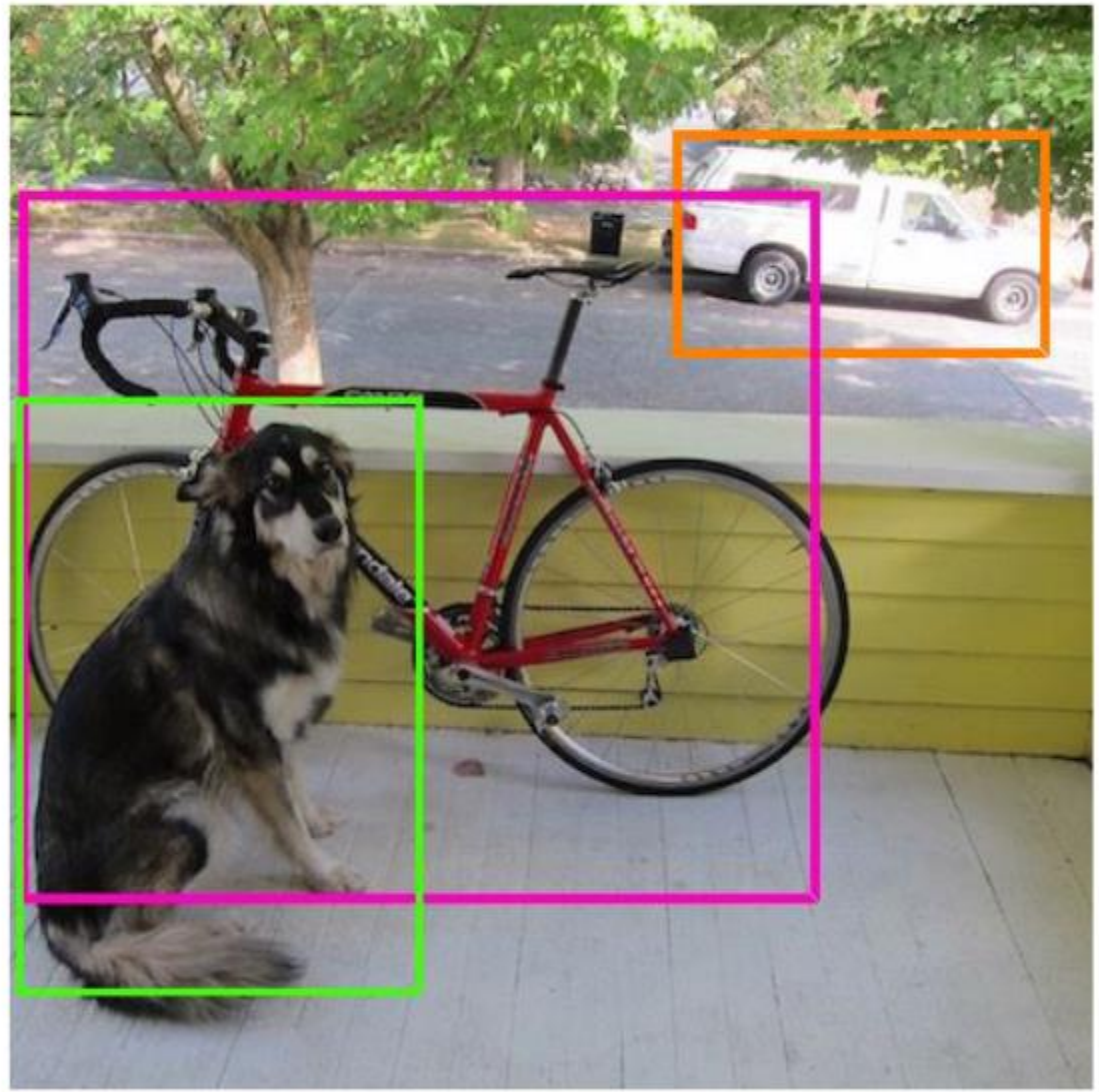
Conditioned on object: P(Car | Object)

Then we combine the box and class predictions.



P(class|Object) * P(Object)
=P(class)

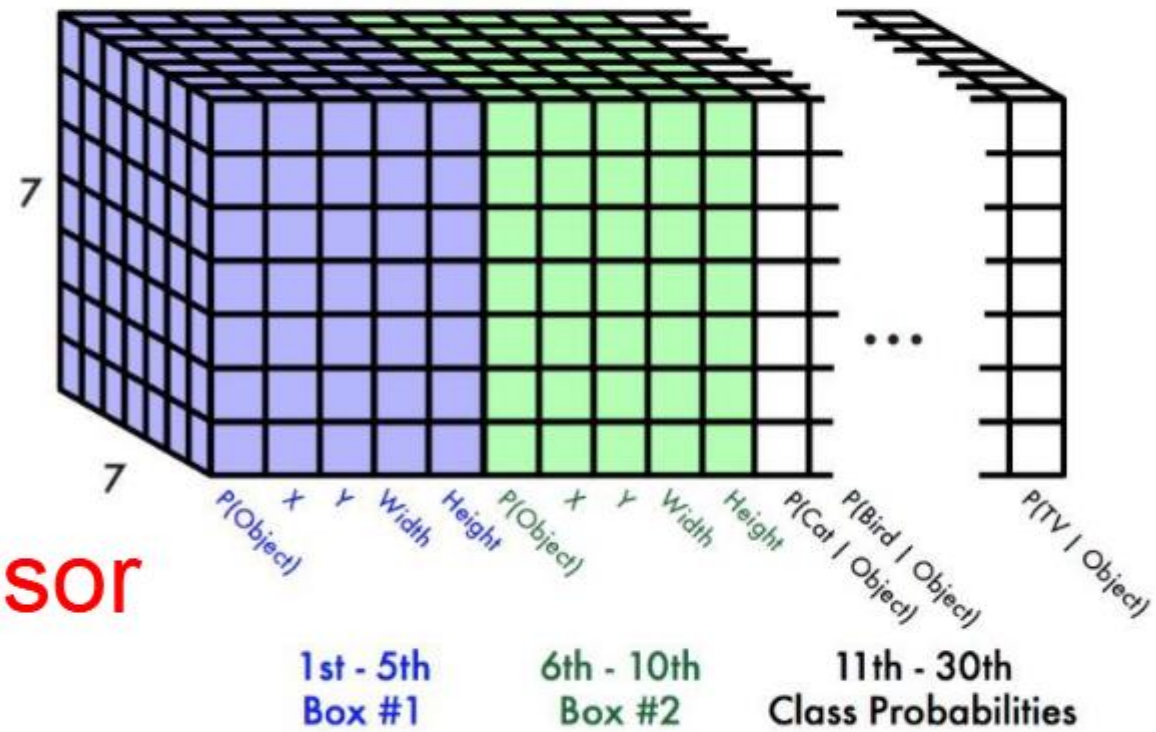
Finally we do threshold detections and NMS



https://docs.google.com/presentation/d/1kAa7NOamBt4calBU9iHgT8a86RRHz9Yz2oh4-GTdX6M/edit#slide=id.g151008b386_0_44

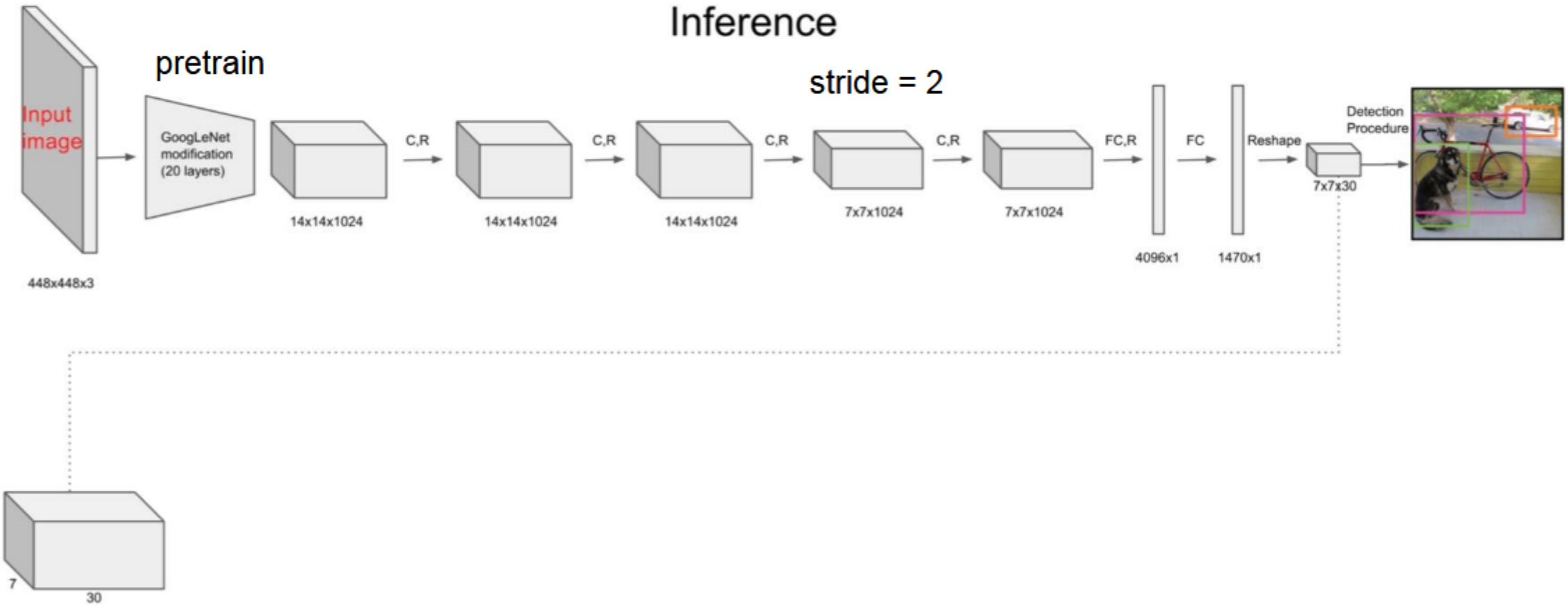
Each cell predicts:

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

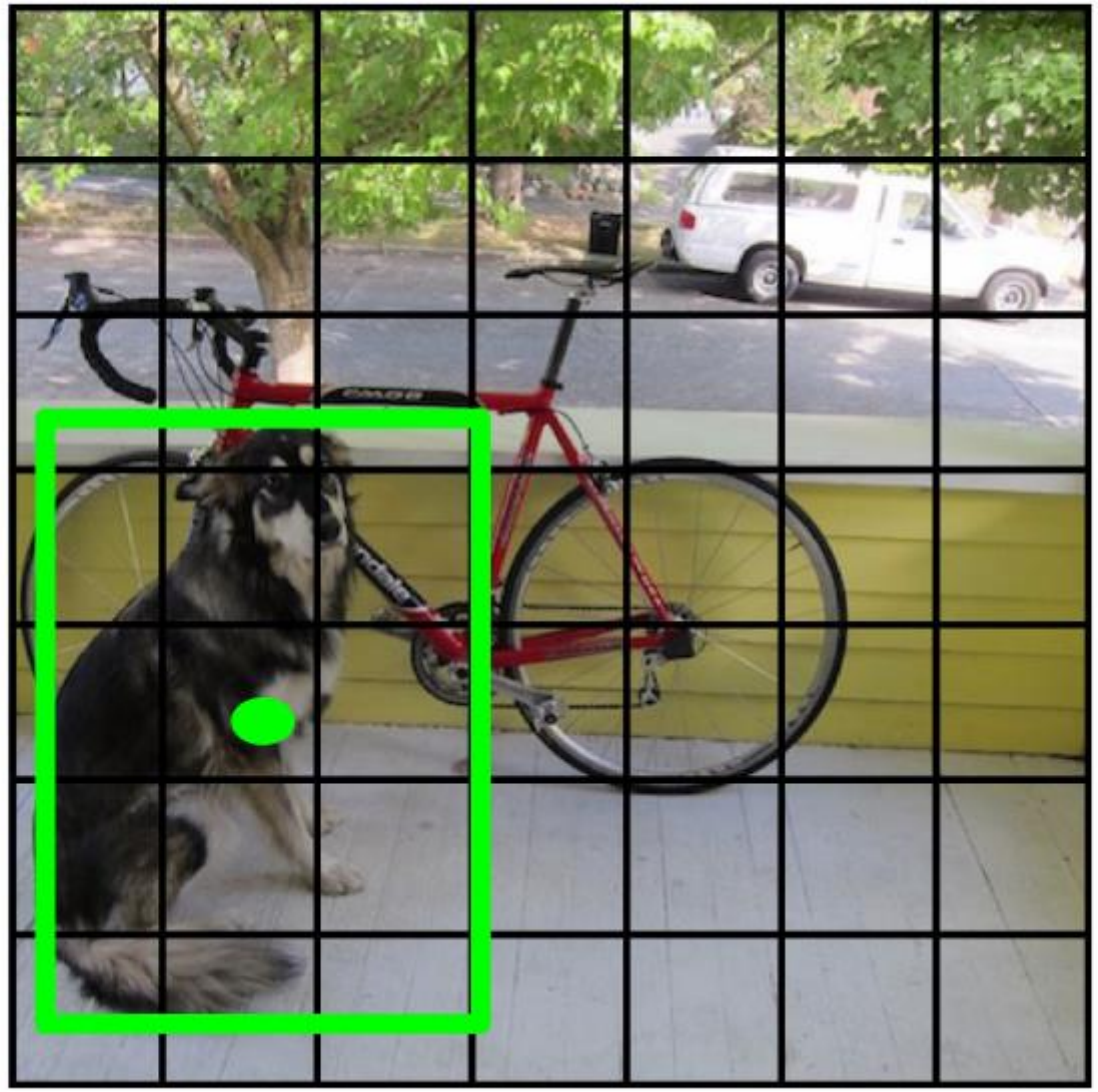


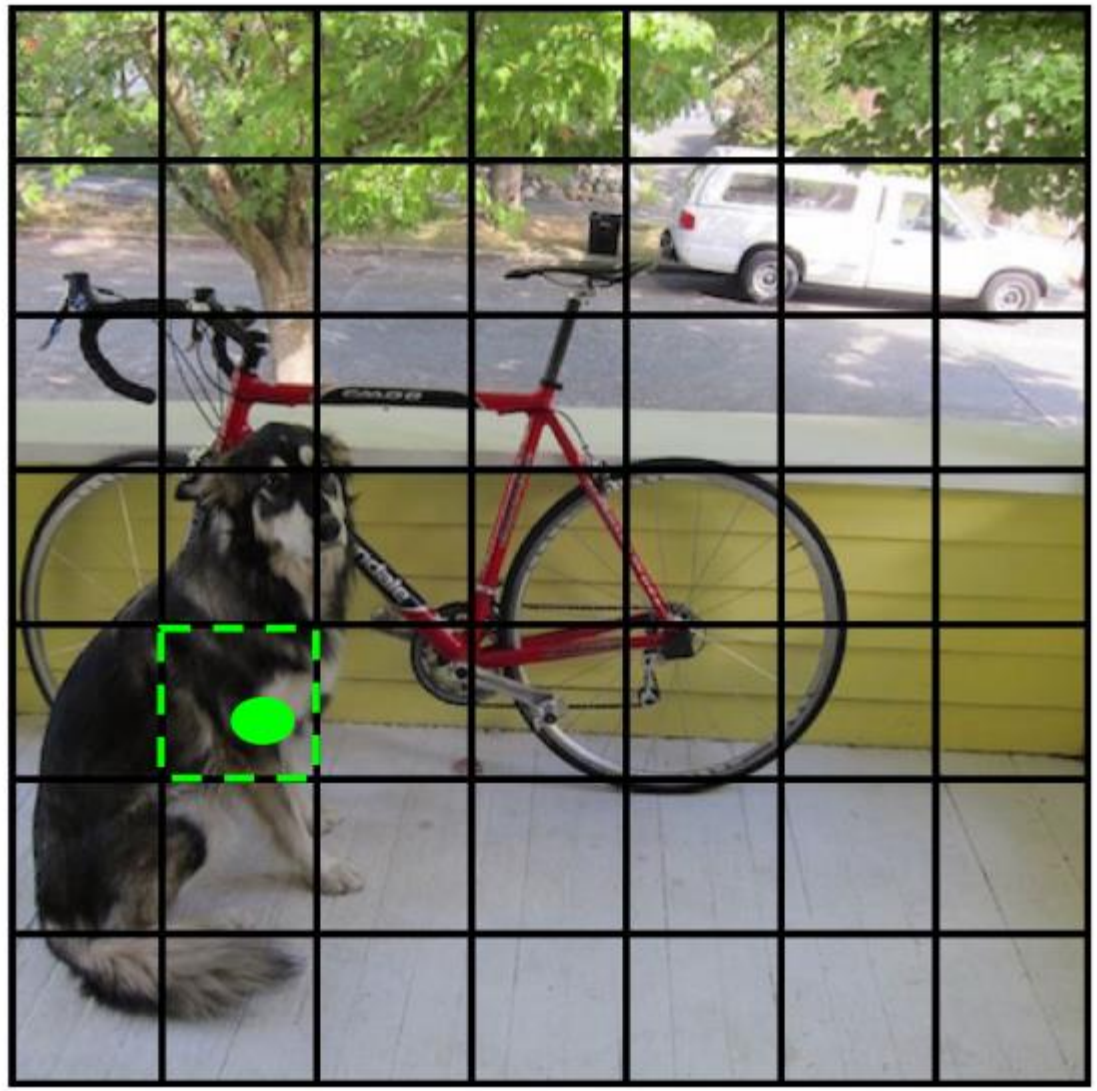
S * S * (B * 5 + C) tensor

https://zhuanlan.zhihu.com/p/24916786?refer=xiaoleimlnote



During training, match example to the right cell

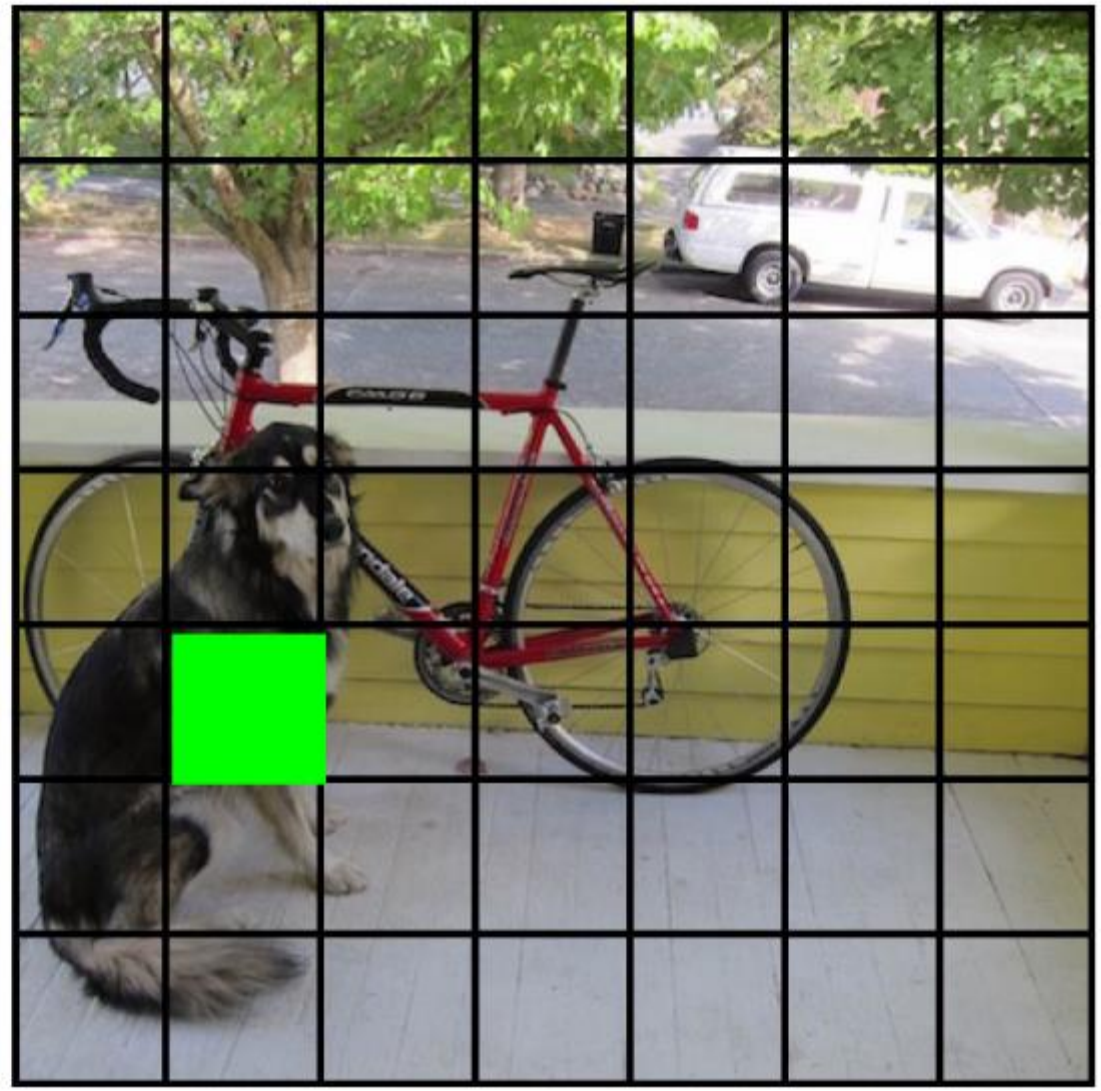




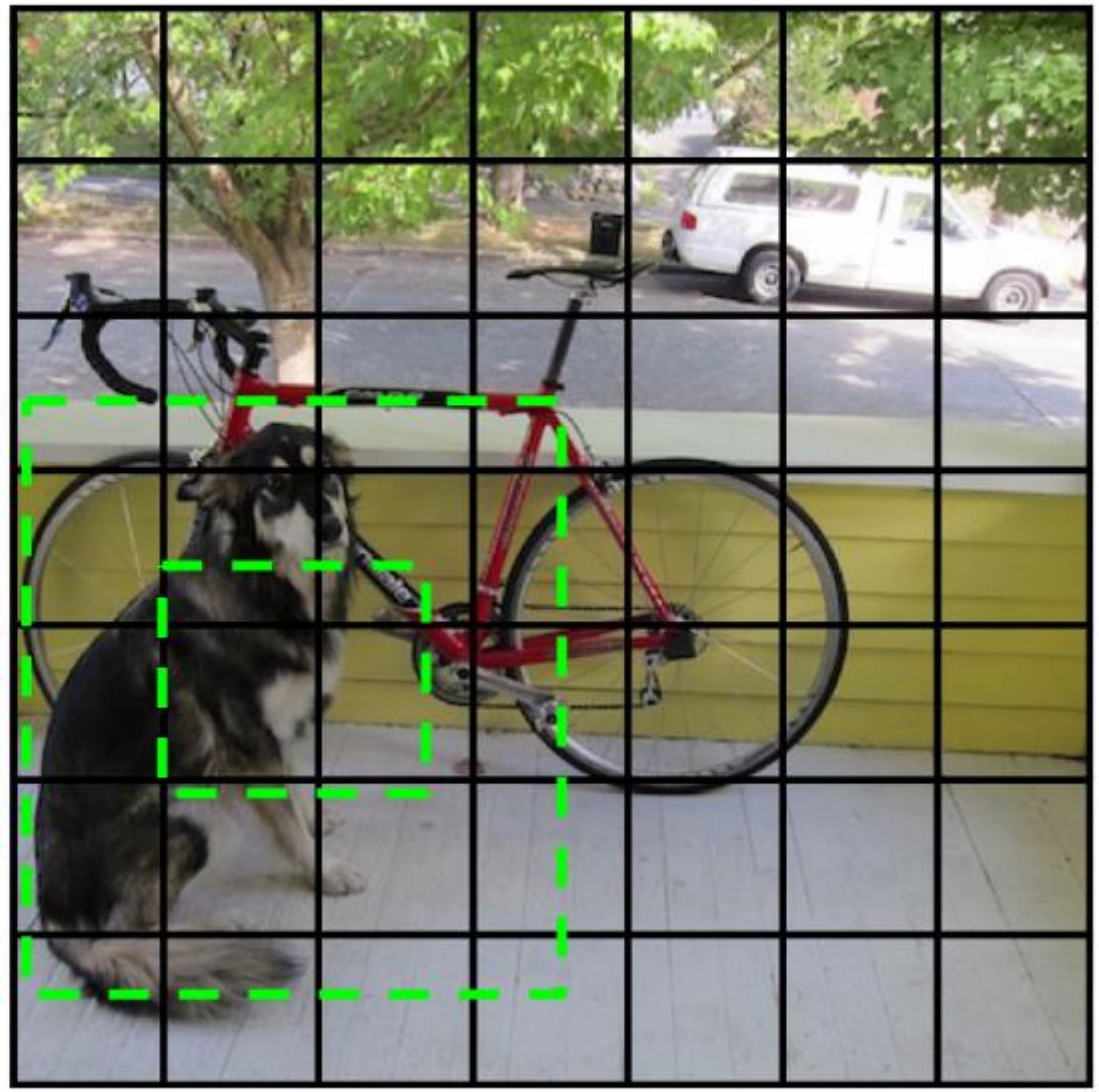
Hence, this cell is responsible for predicting the detection for DOG

Adjust that cell's class prediction

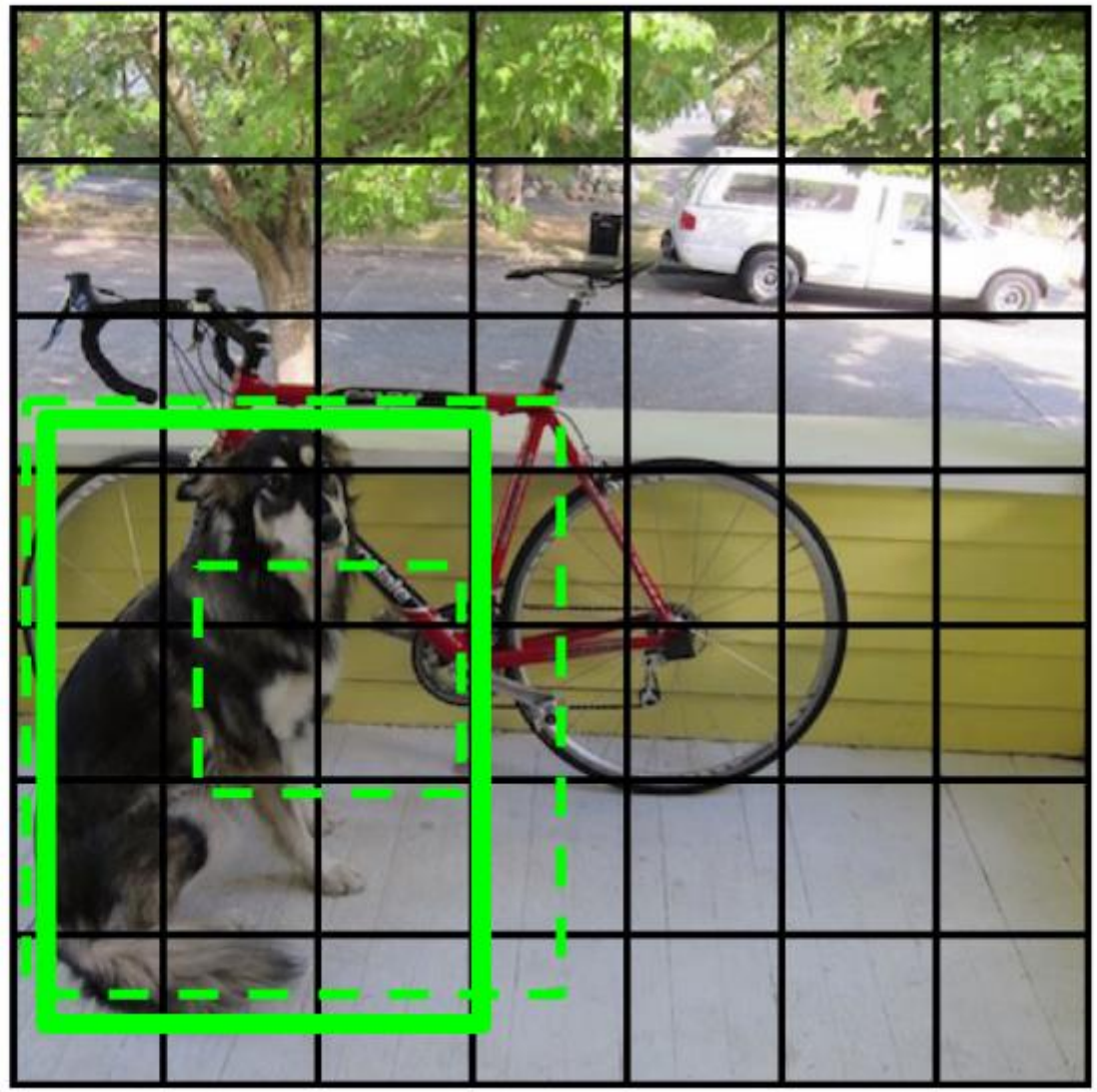
Dog = 1
Cat = 0
Bike = 0
...



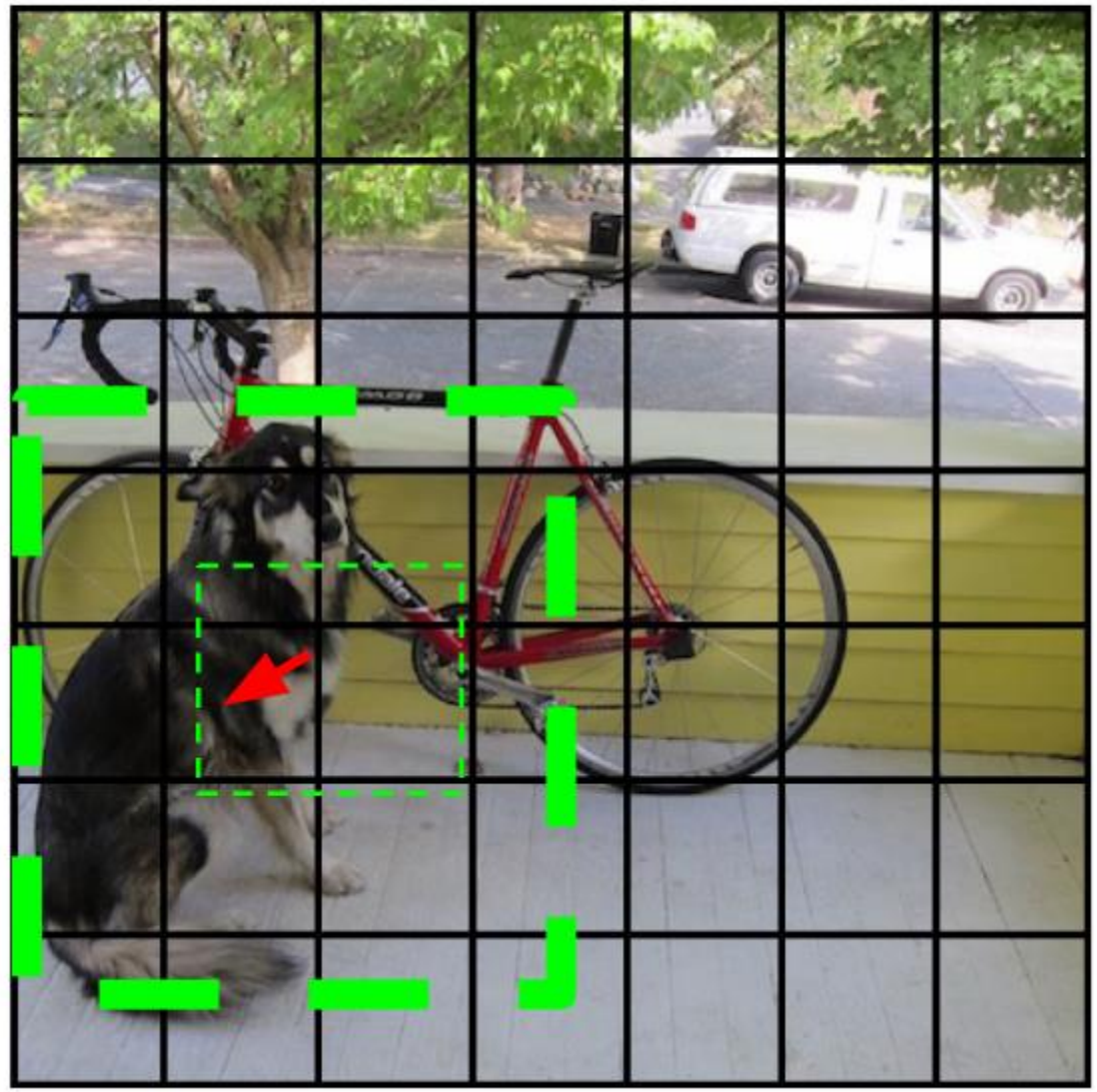
Look at that cell's predicted boxes



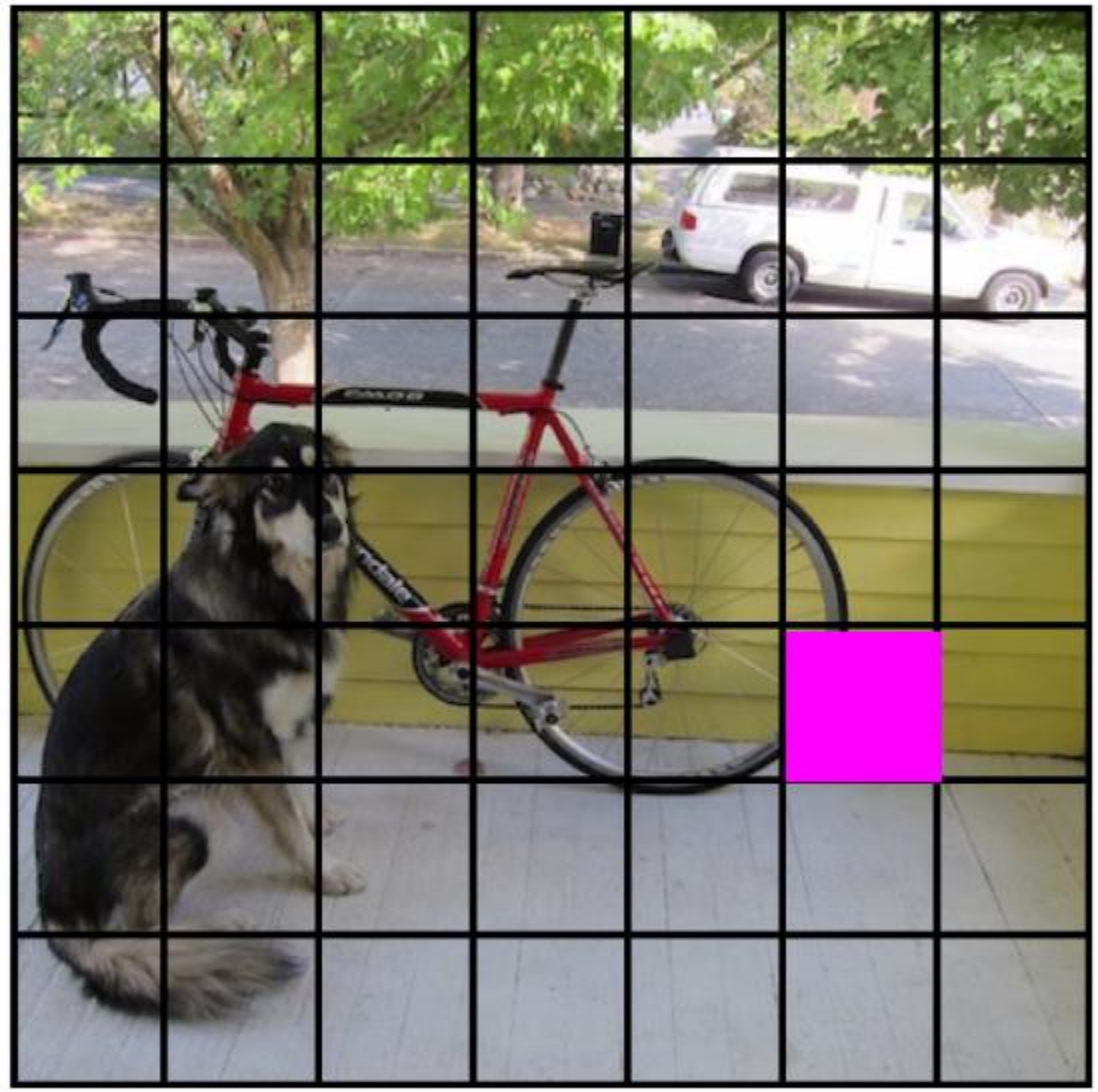
Find the best one, adjust it, increase the confidence



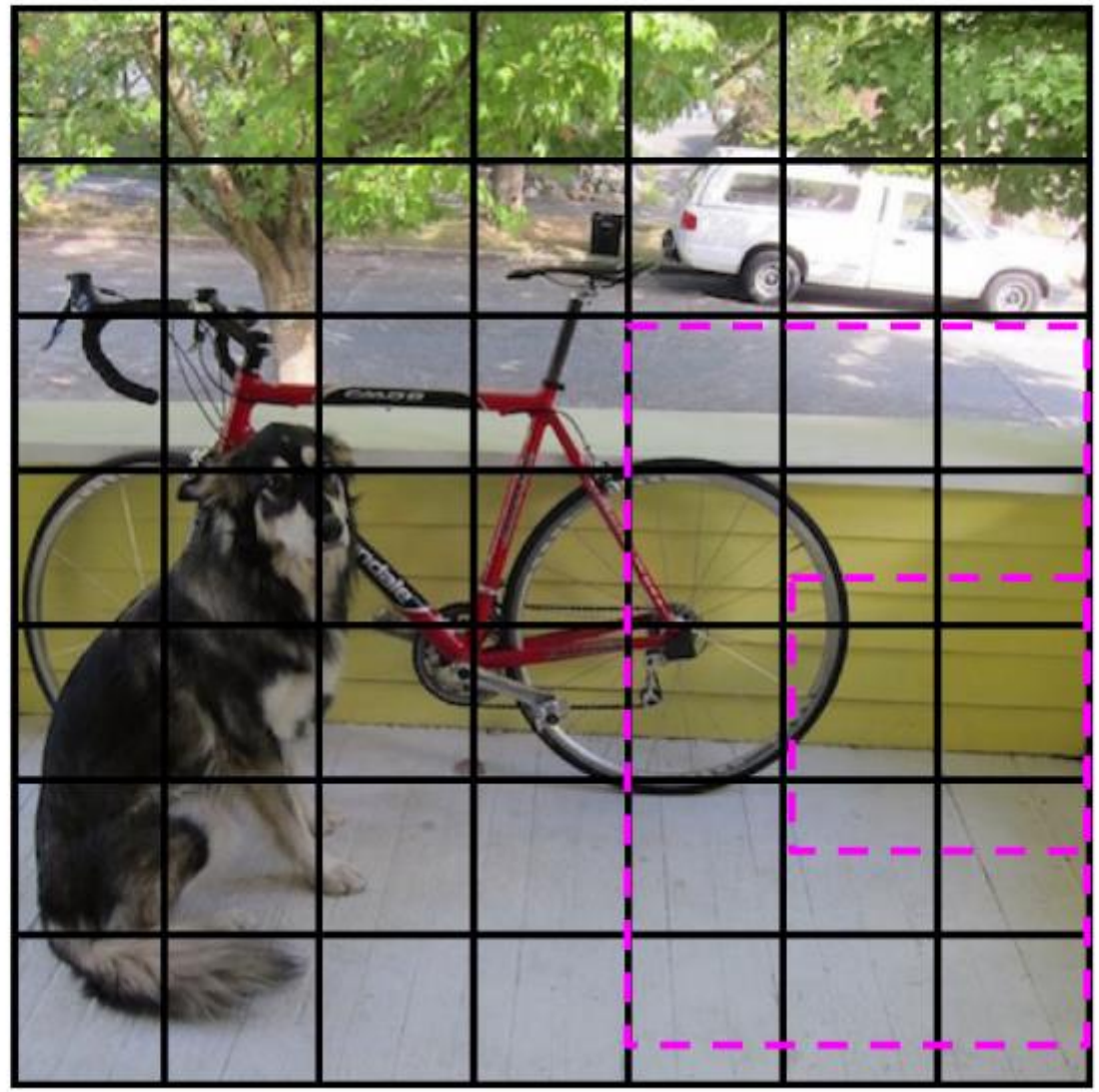
Decrease the confidence of the other box



Some cells don’t have any ground truth detections!



Decrease the confidence of these boxes



Loss Function (sum-squared error)

loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \quad (3)$$

$\mathbb{1}_{ij}^{obj}$ **The *j*th bbox predictor** in ***cell i*** is "responsible" for that prediction

$\mathbb{1}_{ij}^{noobj}$

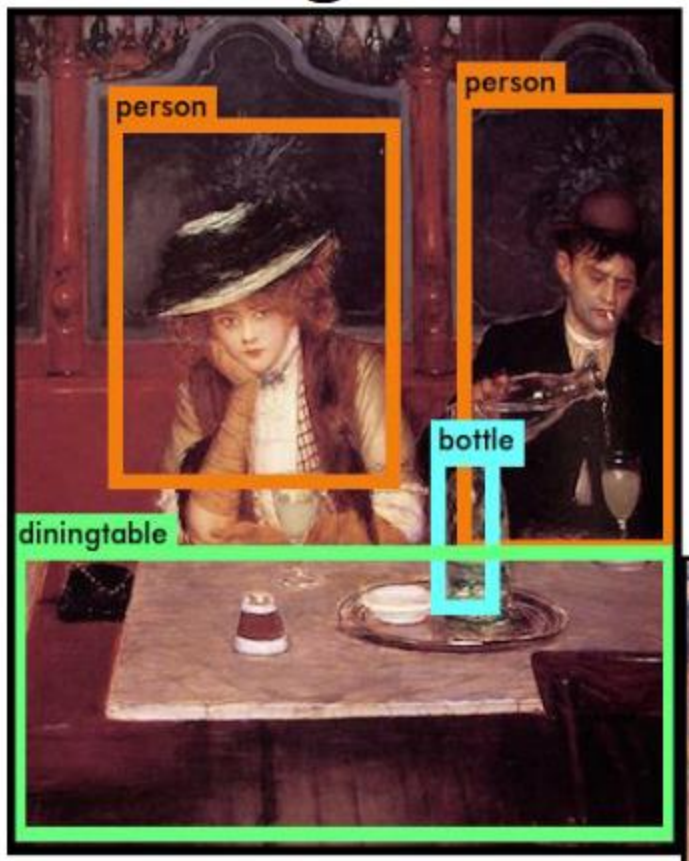
$\mathbb{1}_{i}^{obj}$ If object appears in ***cell i***

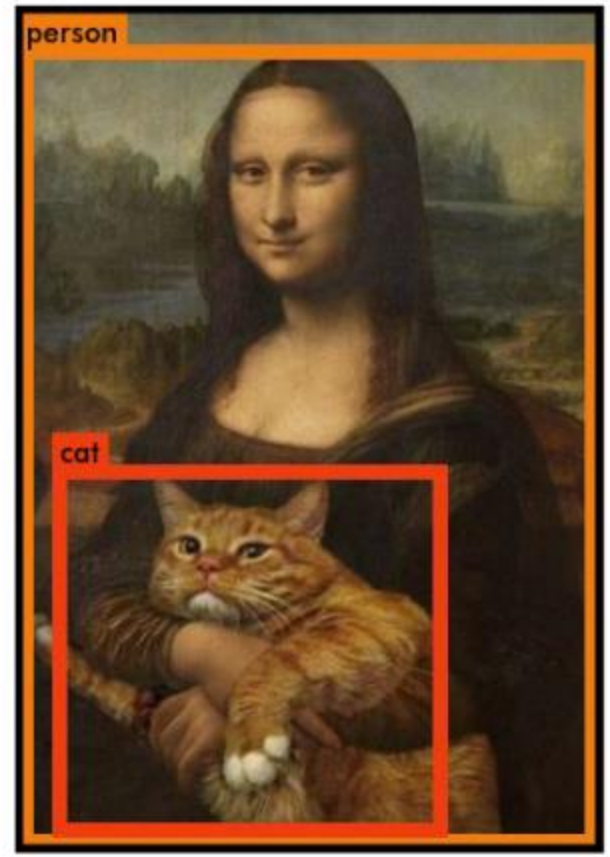
Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is "responsible" for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

Comparisons

	Pascal 2007 mAP	**Speed**	
DPM v5	33.7	.07 FPS	14 s/img
R-CNN	66.0	.05 FPS	20 s/img
Fast R-CNN	70.0	.5 FPS	2 s/img
Faster R-CNN	73.2	7 FPS	140 ms/img
YOLO	63.4	45 FPS	22 ms/img

YOLO generalizes well to new domains (like art)





Ref: https://pjreddie.com/publications/

Evaluations

Confidence score is the probability that an anchor box contains an object. It is usually predicted by a classifier.

Intersection over Union (IoU) is defined as the area of the intersection divided by the area of the union of a predicted bounding box (B_p) and a ground-truth box (B_{gt}):

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (1)$$

Precision and recall

- True positive
- True negative
- False positive
- False negative

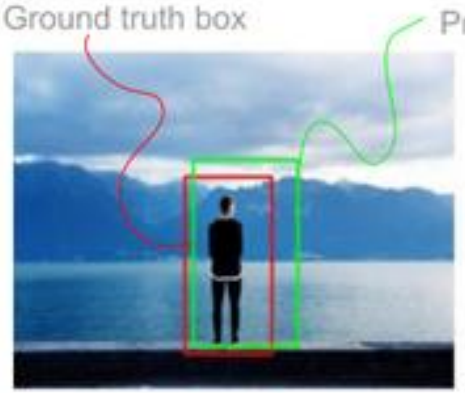
Precision is defined as the number of true positives divided by the sum of true positives and false positives:

$$precision = \frac{TP}{TP + FP} \quad (2)$$

Recall is defined as the number of true positives divided by the sum of true positives and false negatives (note that the sum is just the number of ground-truths, so there's no need to count the number of false negatives):

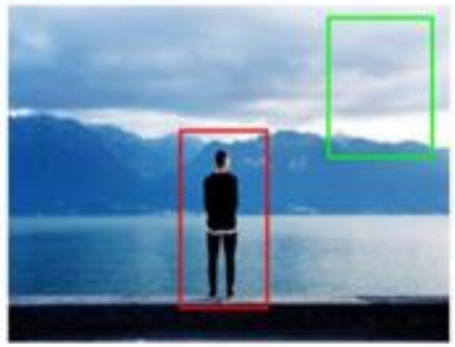
$$recall = \frac{TP}{TP + FN} \quad (3)$$

True Positive - TP



The object **is there**, and the model **detects** it, with an IoU against ground truth box **above** the **threshold**.

False Positive - FP



Left: The object **is there**, but the predicted box has an IoU against ground truth box **less** than **threshold**.

Right: The object is **not there**, and the model **detects** one.

False Negative - FN

The object **is there**, and the model **doesn't** detect it. The ground truth object has **no** prediction.

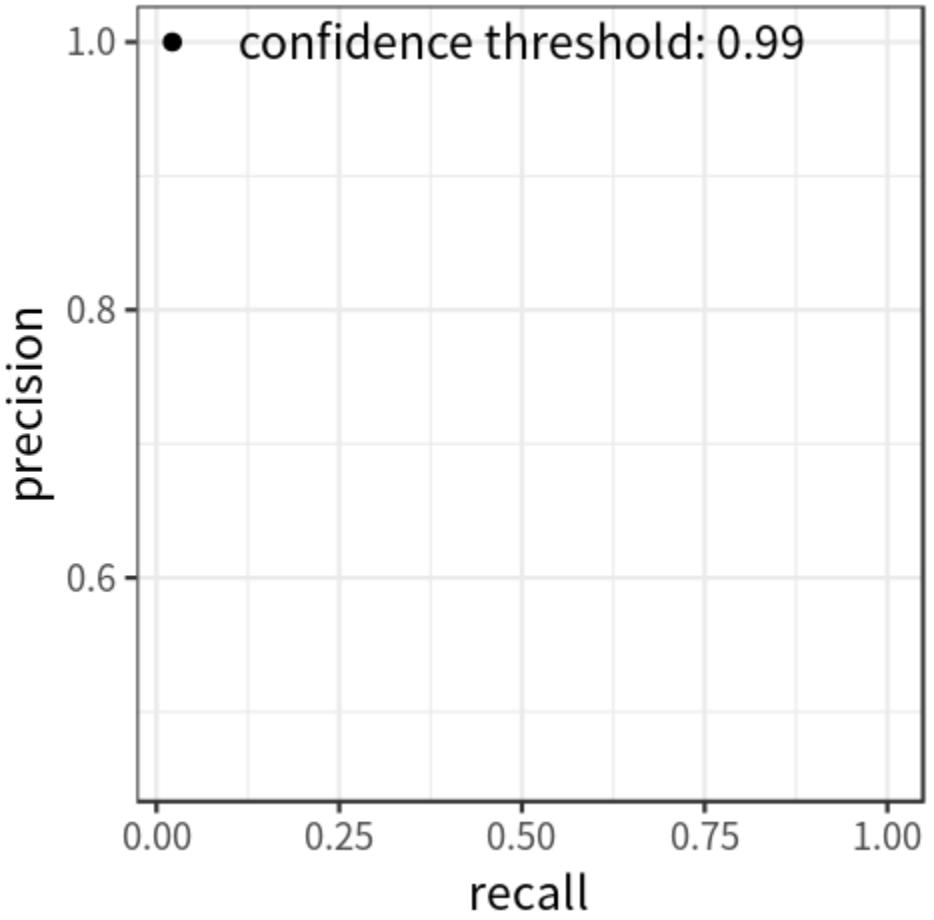
Some interpretations

- High recall but low precision implies that all ground truth objects have been detected, but most detections are incorrect (many false positives).
- Low recall but High precision implies that all predicted boxes are correct, but most ground truth objects have been missed (many false negatives).
- High precision and high recall, the ideal detector has most ground truth objects detected correctly.

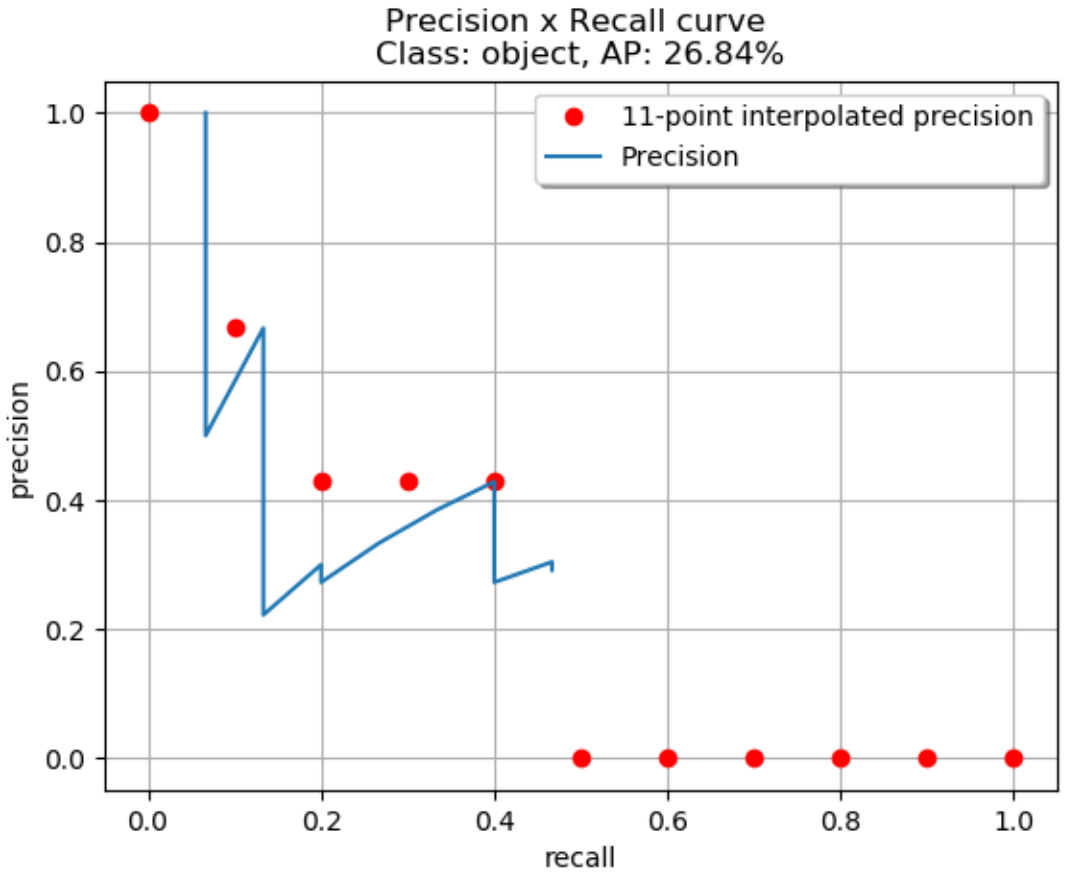
Some interpretations

- When multiple boxes detect the same object, the box with the highest IoU is considered TP, while the remaining boxes are considered FP.
- If the object is present and the predicted box has an IoU < threshold with ground truth box, The prediction is considered FP. More importantly, because no box detected it properly, the class object receives FN, .
- If the object is not in the image, yet the model detects one then the prediction is considered FP.

PR curve, for different confidence



Average Precision and mean AP (for N classes)



$$AP = \frac{1}{11} \sum_{r \in \{0, 0.1, 0.2, \ldots, 1\}} p_{interp}(r)$$

where

$$p_{interp}(r) = \max_{\tilde{r} > r} p(\tilde{r})$$

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i$$

If all points are taken, AP is also called the AUC