Lecture 10:

Optimizing Object Detection:
A Case Study of R-CNN, *Fast* R-CNN, and *Faster* R-CNN

Visual Computing Systems
CMU 15-769, Fall 2016

Today's task: object detection

Image classification: what is the object in this image?

tennis racket

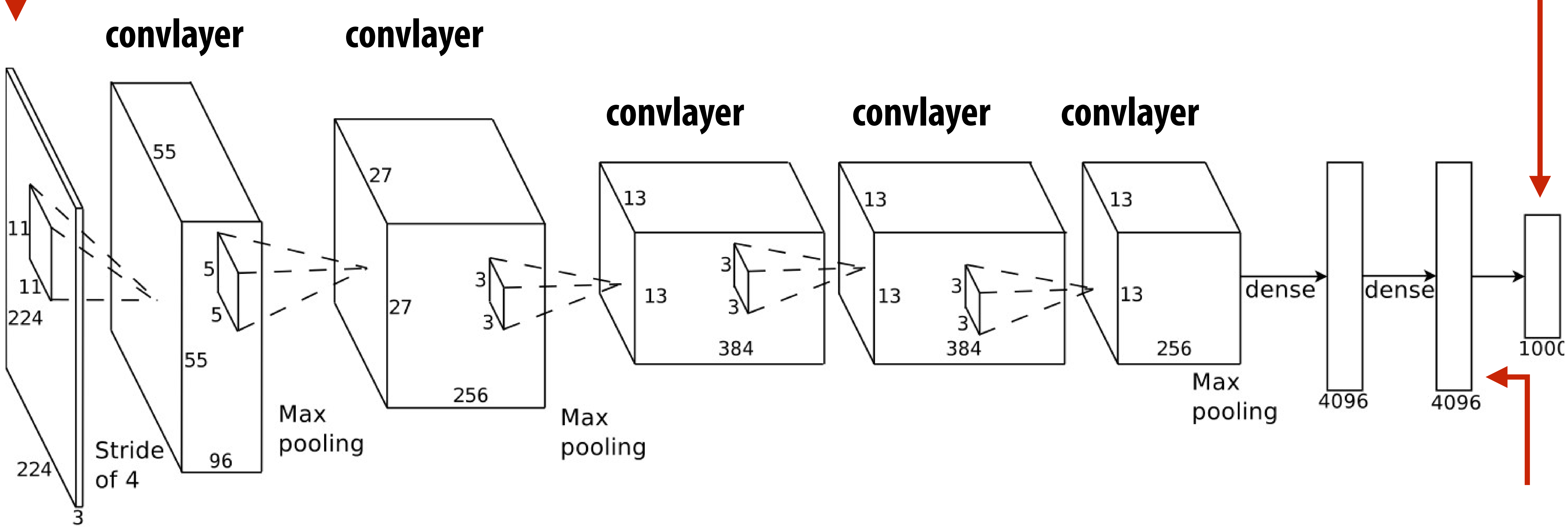
Object detection: where is the tennis racket in this image? (if there is one at all?)

Krizhevsky (AlexNet) image classification network

[Krizhevsky 2012]

Input:
fixed size image

Output:
probability of label
(for 1000 class labels)

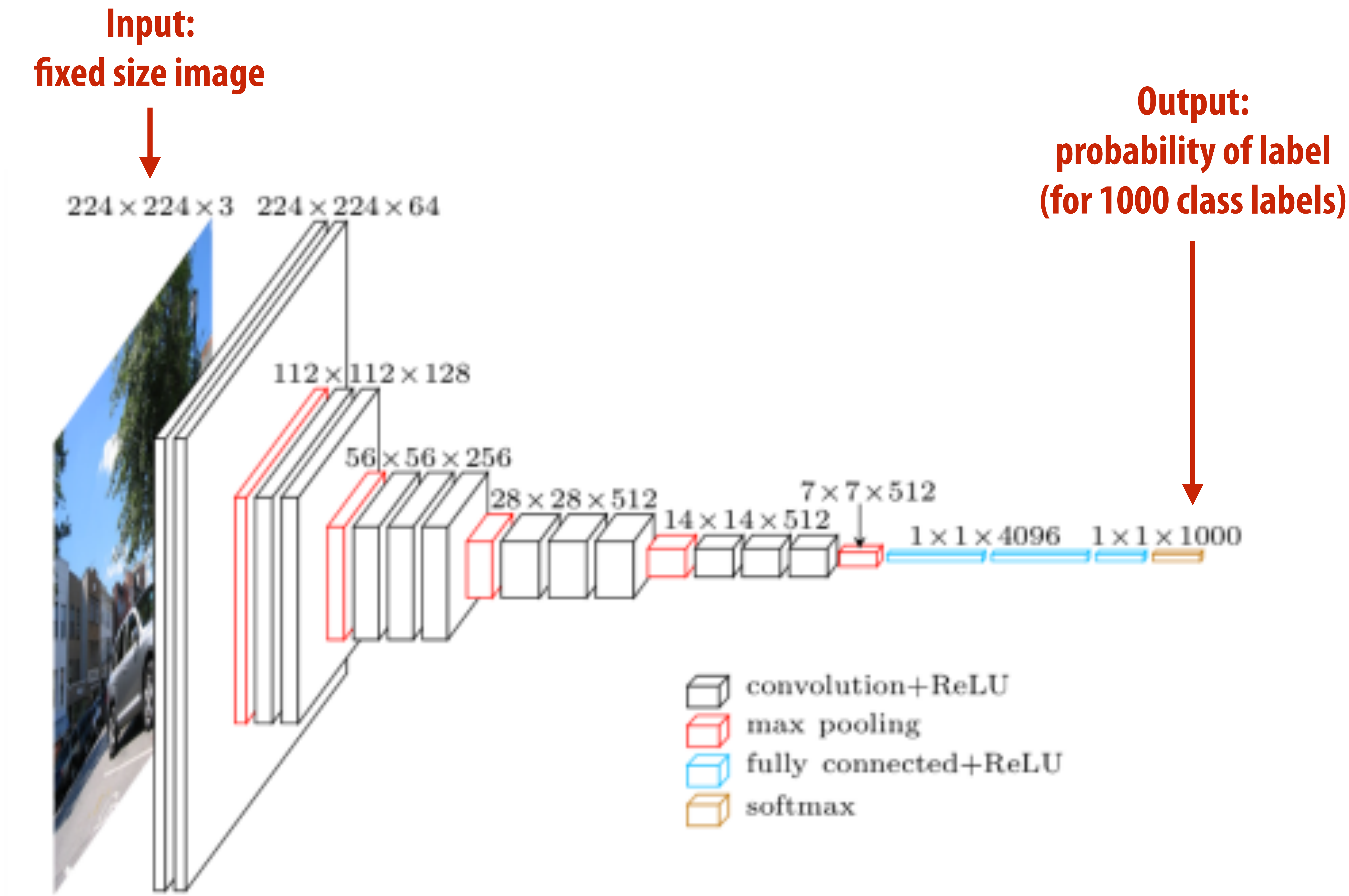


DNN produces feature vector in 4K-dim space that is input to multi-way classifier ("softmax") to produce per-label probabilities

Network assigns input image one of 1000 potential labels.

VGG-16 image classification network

[Simonyan 2015]



Network assigns input image one of 1000 potential labels.

Today: three object detection papers

- R-CNN [Girshick 2014]
- Fast R-CNN [Girshick 2015]
- Faster R-CNN [Ren, He, Girshick, Sun 2015]

- Each paper improves on ***both*** the wall-clock performance and the detection accuracy of the previous

Using AlexNet as a “subroutine in object detection

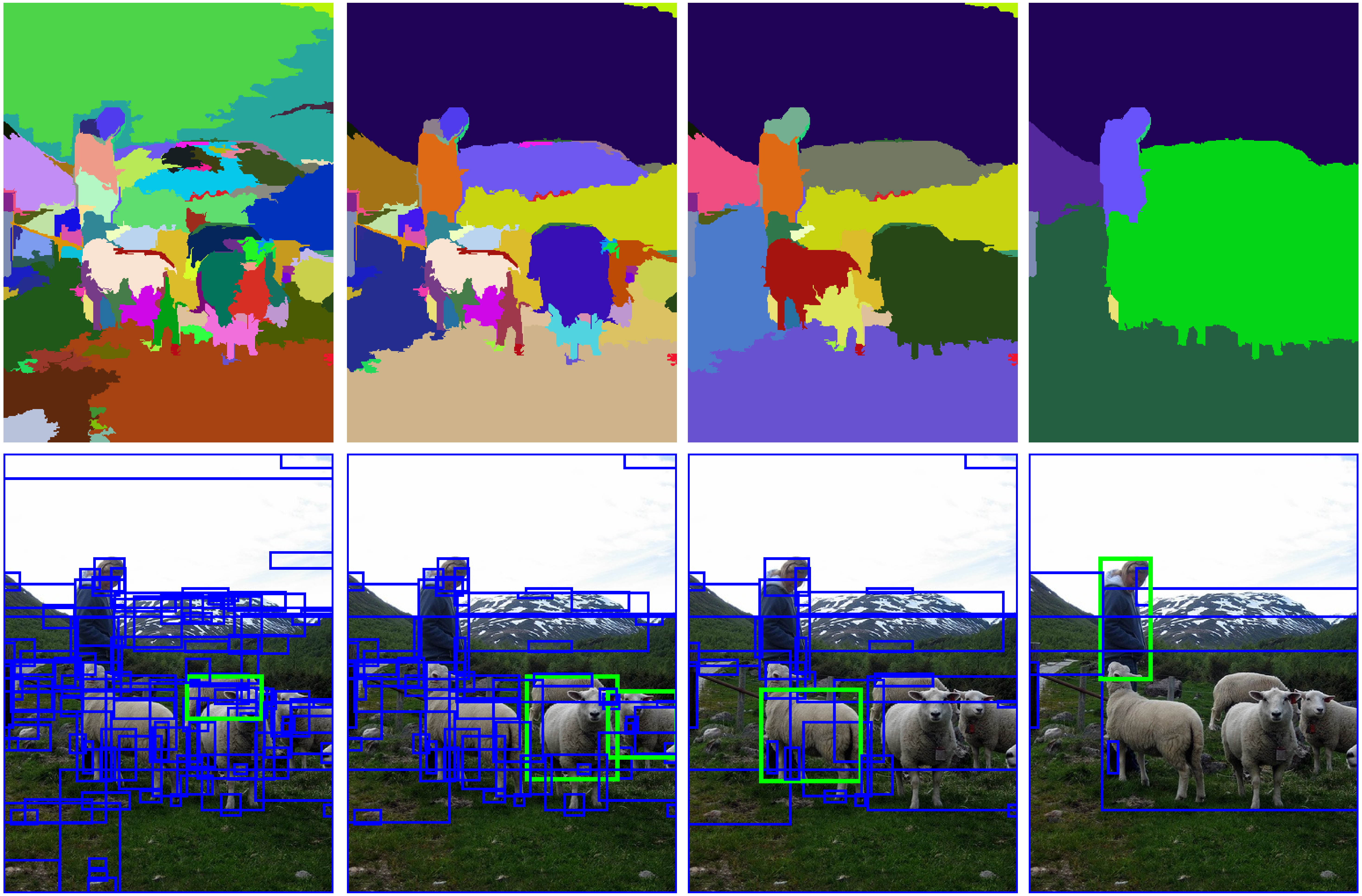
[Girshick 2014]

Search over all regions of the image for objects.
(“Sliding window” over image, repeated for multiple potential object scales)

```
for all region top-left positions (x,y):
   for all region sizes (w,h):
      cropped = image_crop(image, bbox(x,y,w,h))
      resized = image_resize(227,227)
      label = detect_object(resized)
      if (label != background)
         // region defined by bbox(x,y,w,h) contains object
         // of class ‘label’
```

Optimization 1: filter detection work via object proposals

Selective search [Uijlings IJCV 2013]



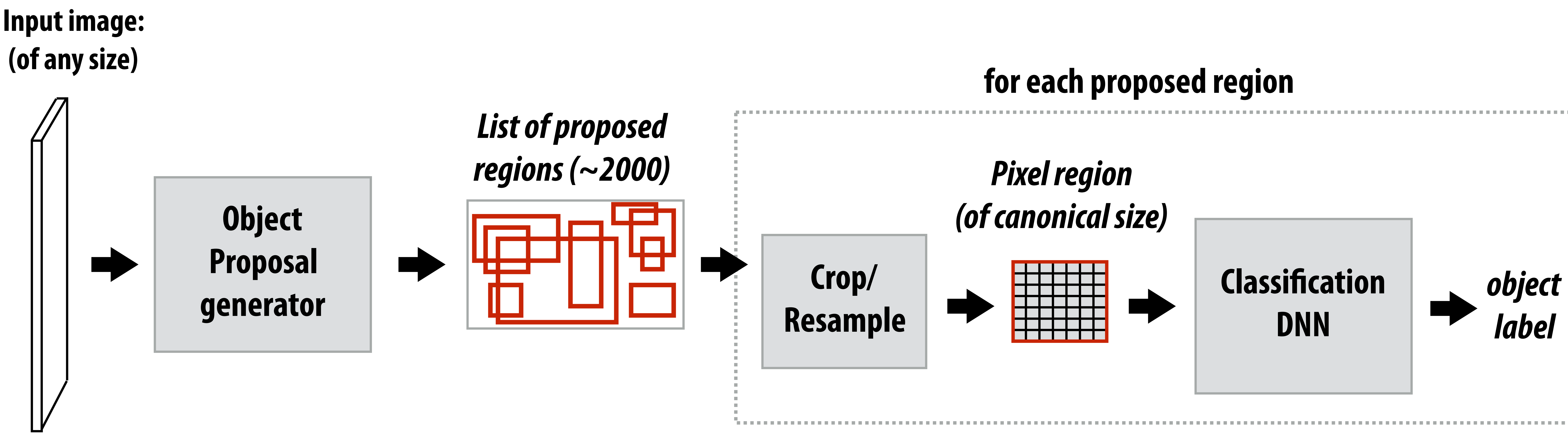
Input: image

Output: list of regions (various scales) that are likely to contain objects

Idea: proposal algorithm filters parts of the image not likely to contain objects

Object detection pipeline executed only on proposed regions

[Girshick 2014]

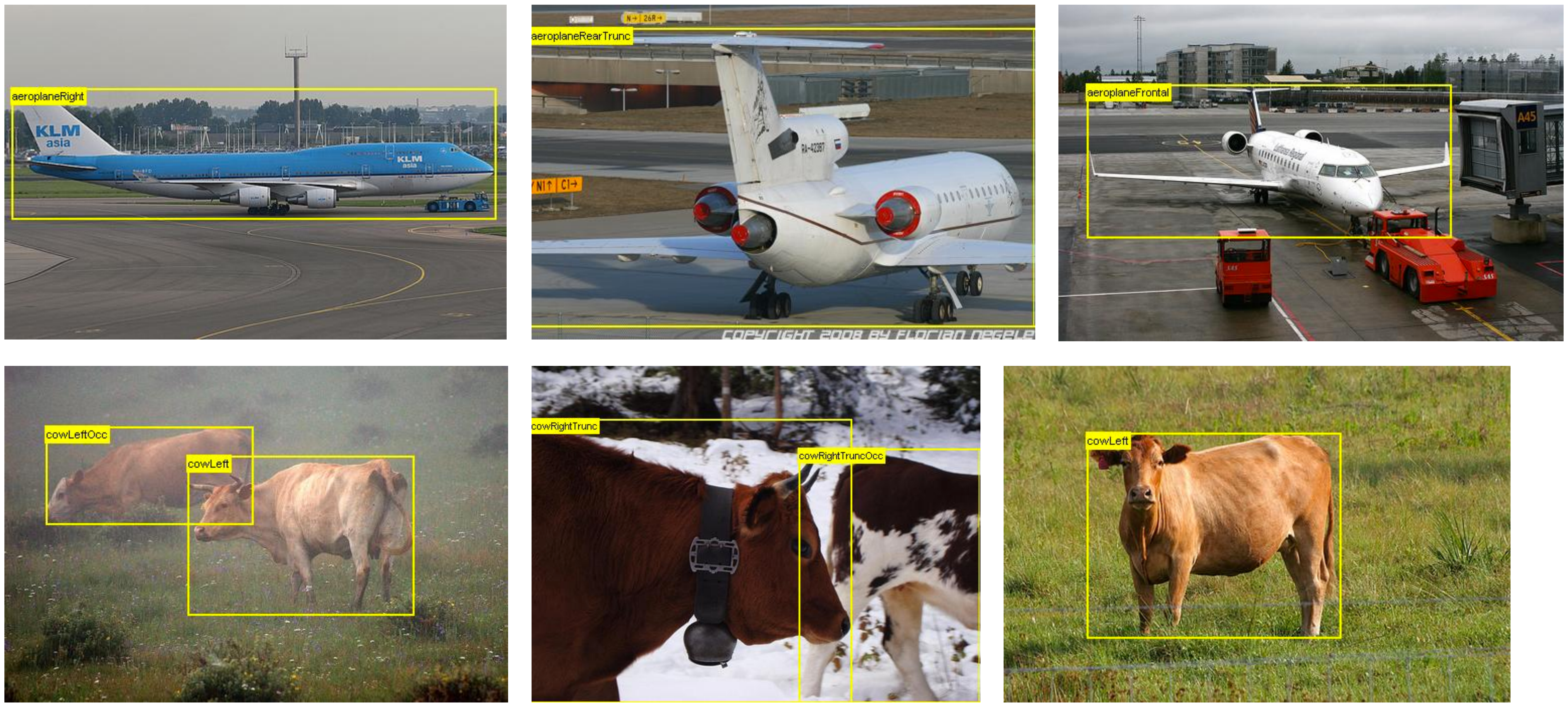


Object detection performance on Pascal VOC

Example training data

airplanes

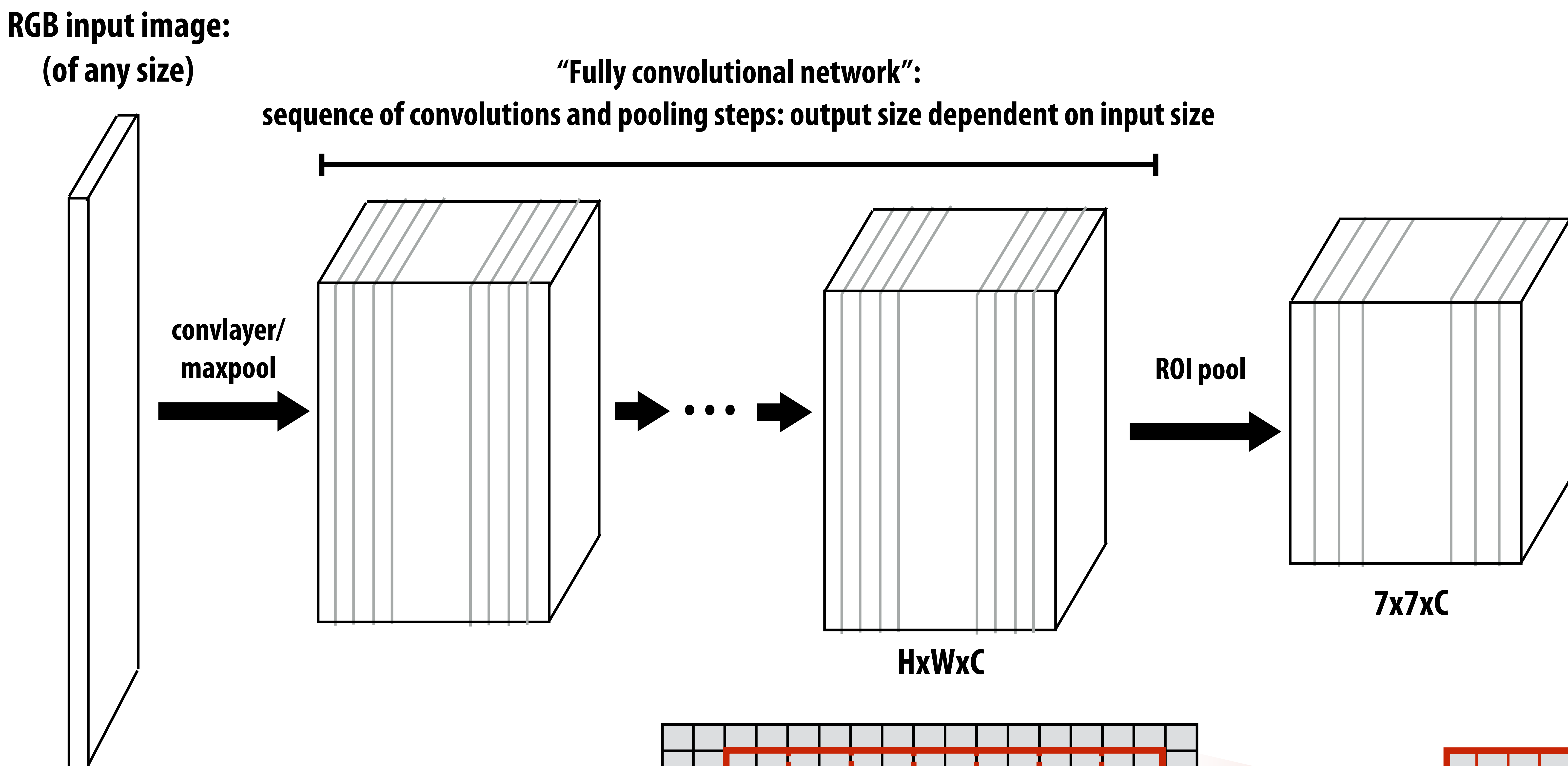
cow

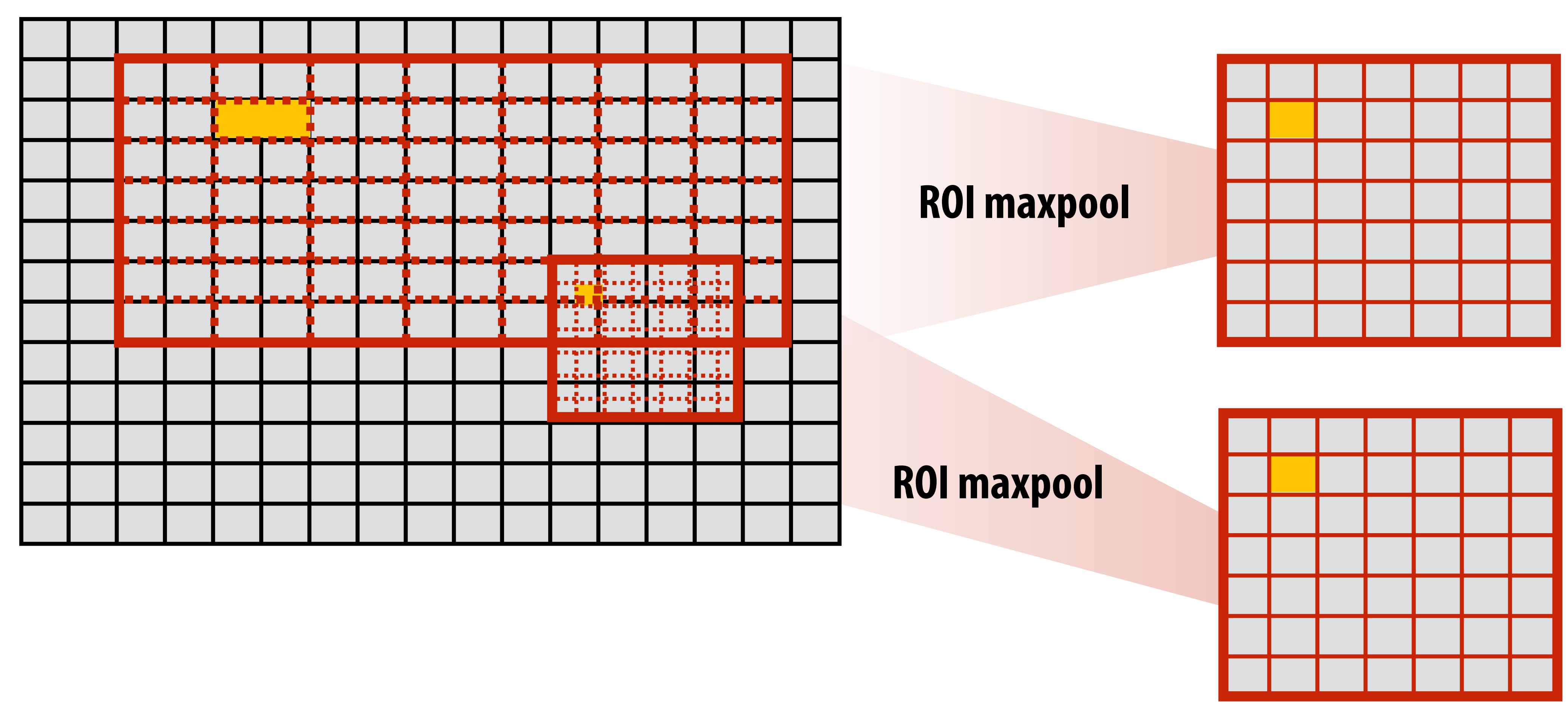


VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [18]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [34]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [36]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [16]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2

"Fine tuned" DNN weights obtained by "pretraining" for object classification on ImageNet for the 20 VOC categories (+ 1 "background" category)

Optimization 2: region of interest pooling





Idea: the output of early convolutional layers of network on downsampled input region is approximated by resampling output of fully-convolutional implementation of conv layers.

Performance optimization: can evaluate convolutional layers once on large input, then reuse intermediate output many times to approximate response of a subregion of image.

Optimization 2: region of interest pooling

Form of "approximate common subexpression elimination"

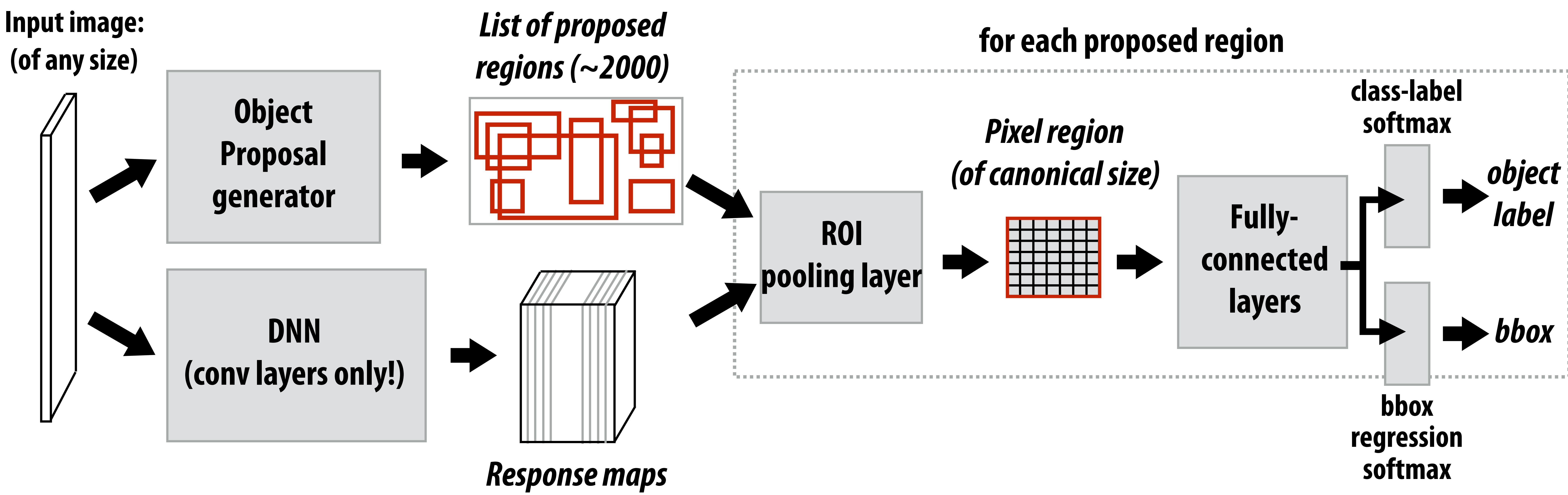
```
for all proposed regions (x,y,w,h):   // 1000's regions/image
   cropped = image_crop(image, bbox(x,y,w,h))
   resized = image_resize(227,227)
   label = detect_object(resized)
```

redundant work (many regions overlap, so responses at lower network layers are computed many times)

```
conv5_response = evaluate_conv_layers(image)
for all proposed regions (x,y,w,h):
   region_conv5 = roi_pool(conv5_response, bbox(x,y,w,h))
   label = evaluate_fully_connected_layers(region_conv5)
```

computed once per image

Fast R-CNN pipeline [Girshick 2015]



Evaluation speed: 146x faster than R-CNN (47sec/img → 0.32 sec/img)

[This number excludes cost of proposals]

Training speed: 9x faster than R-CNN

Training mini-batch: pick N images, pick 128/N boxes from each image (allows sharing of conv-layer pre-computation for multiple image-box training samples)

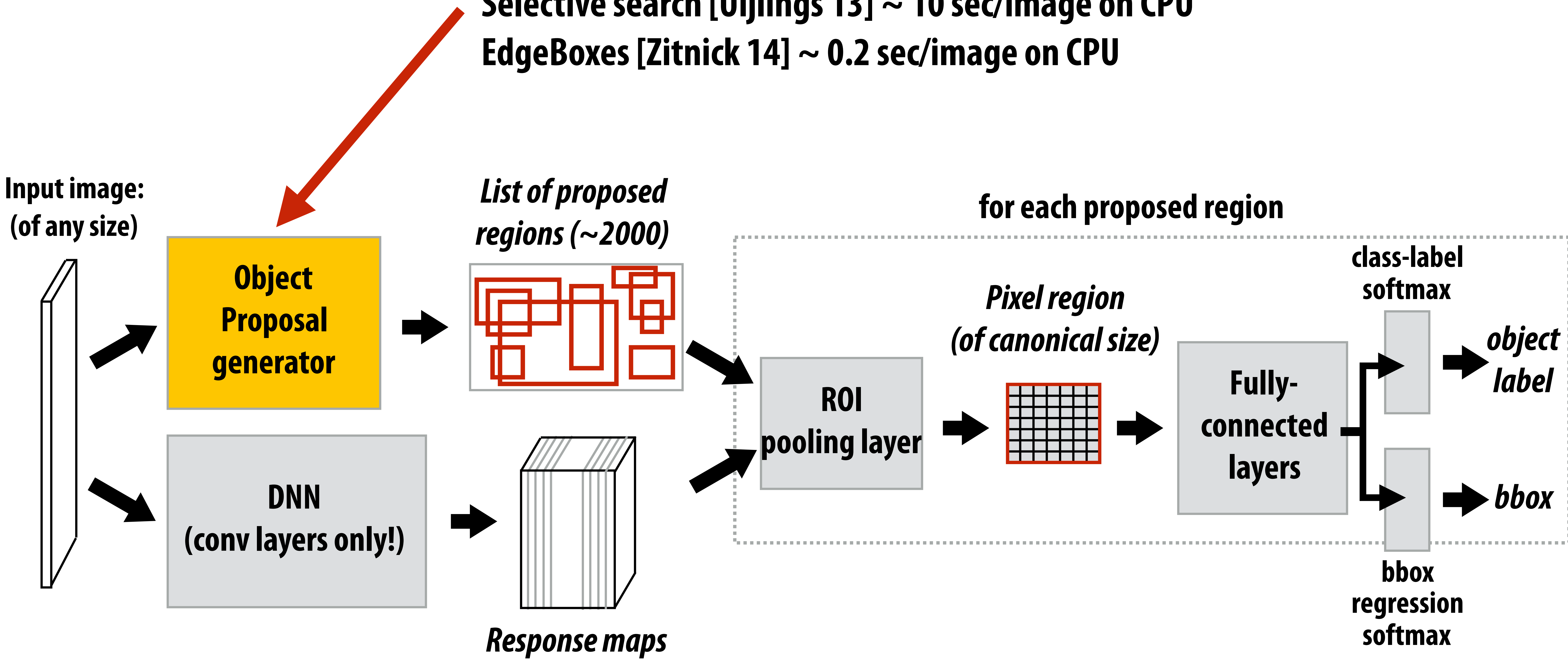
Simultaneously train class predictions and bbox predictions: joint loss = class label loss + bbox loss

Note: training updates weights in BOTH fully connected/softmax layers AND cons layers

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1

Problem: bottleneck is now generating proposals

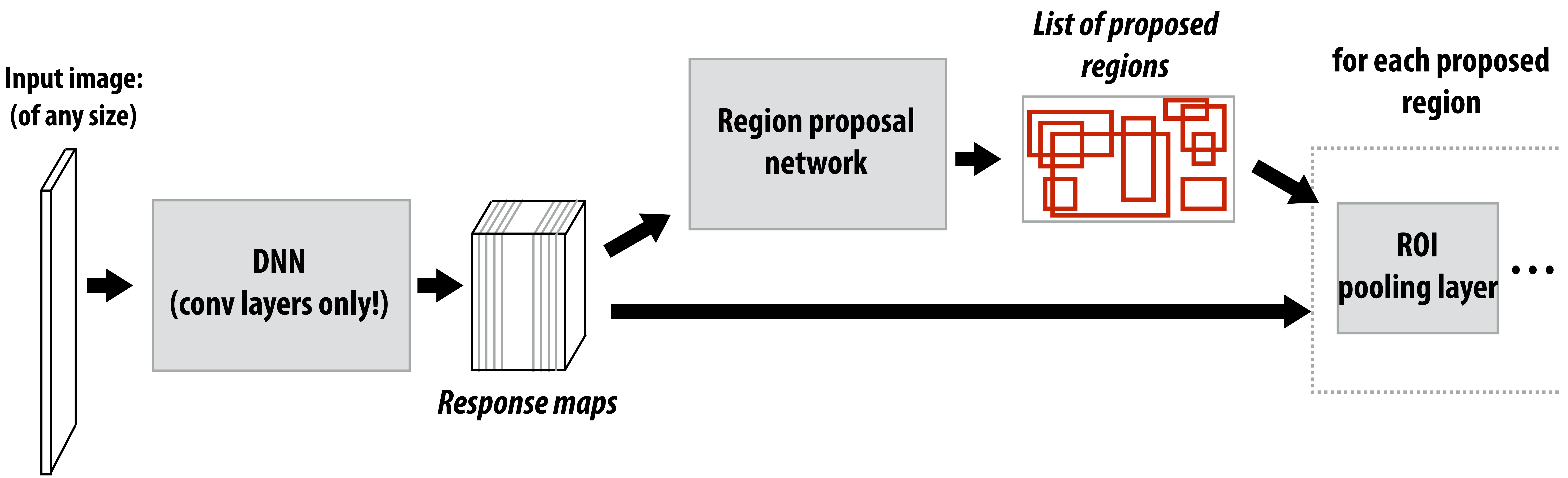
Selective search [Uijlings 13] ~ 10 sec/image on CPU
EdgeBoxes [Zitnick 14] ~ 0.2 sec/image on CPU



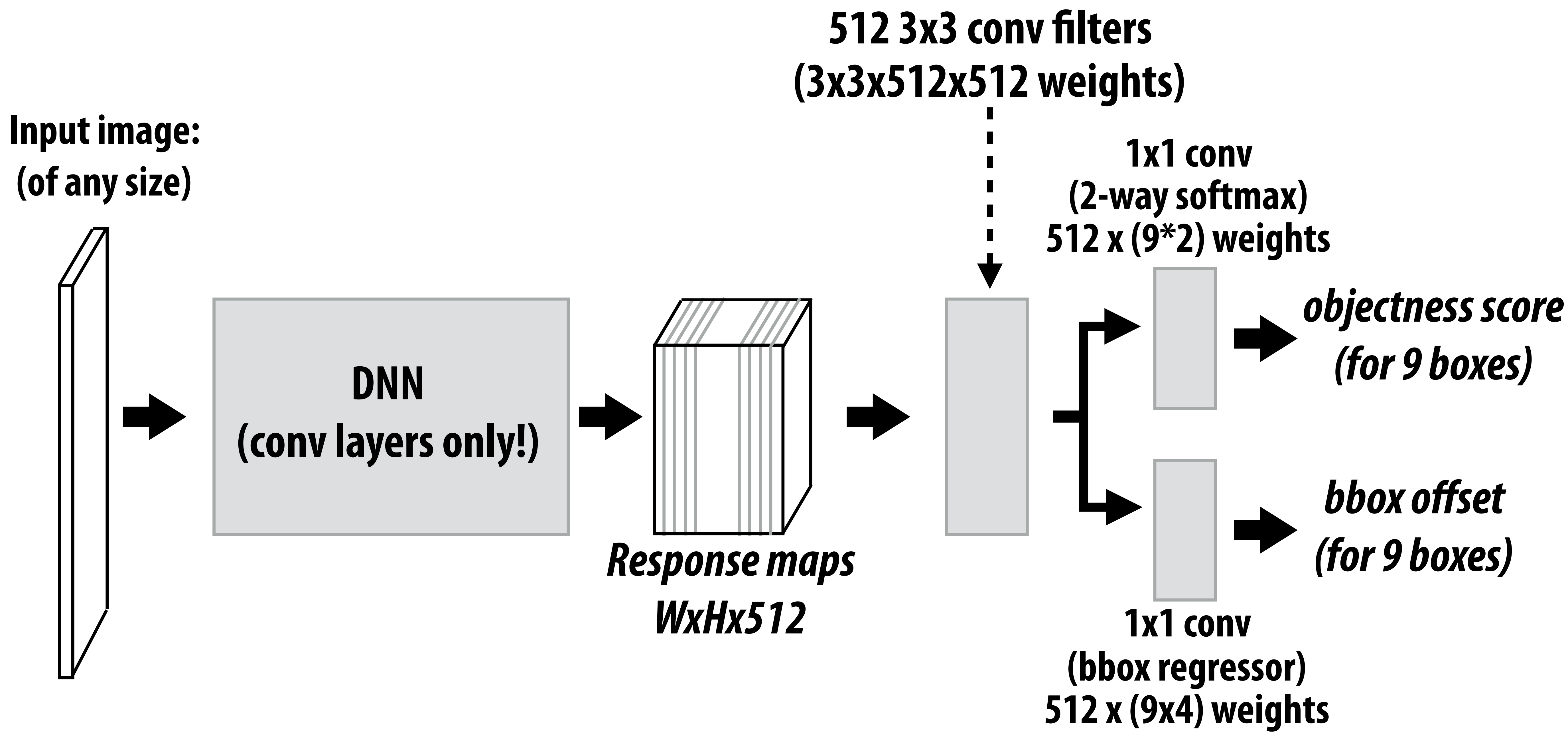
Idea: why not predict regions from the convolutional feature maps that must be computed for detection anyway? (share computation between proposals and detection)

Faster R-CNN using a region proposal network (RPN)

[Ren 2015]



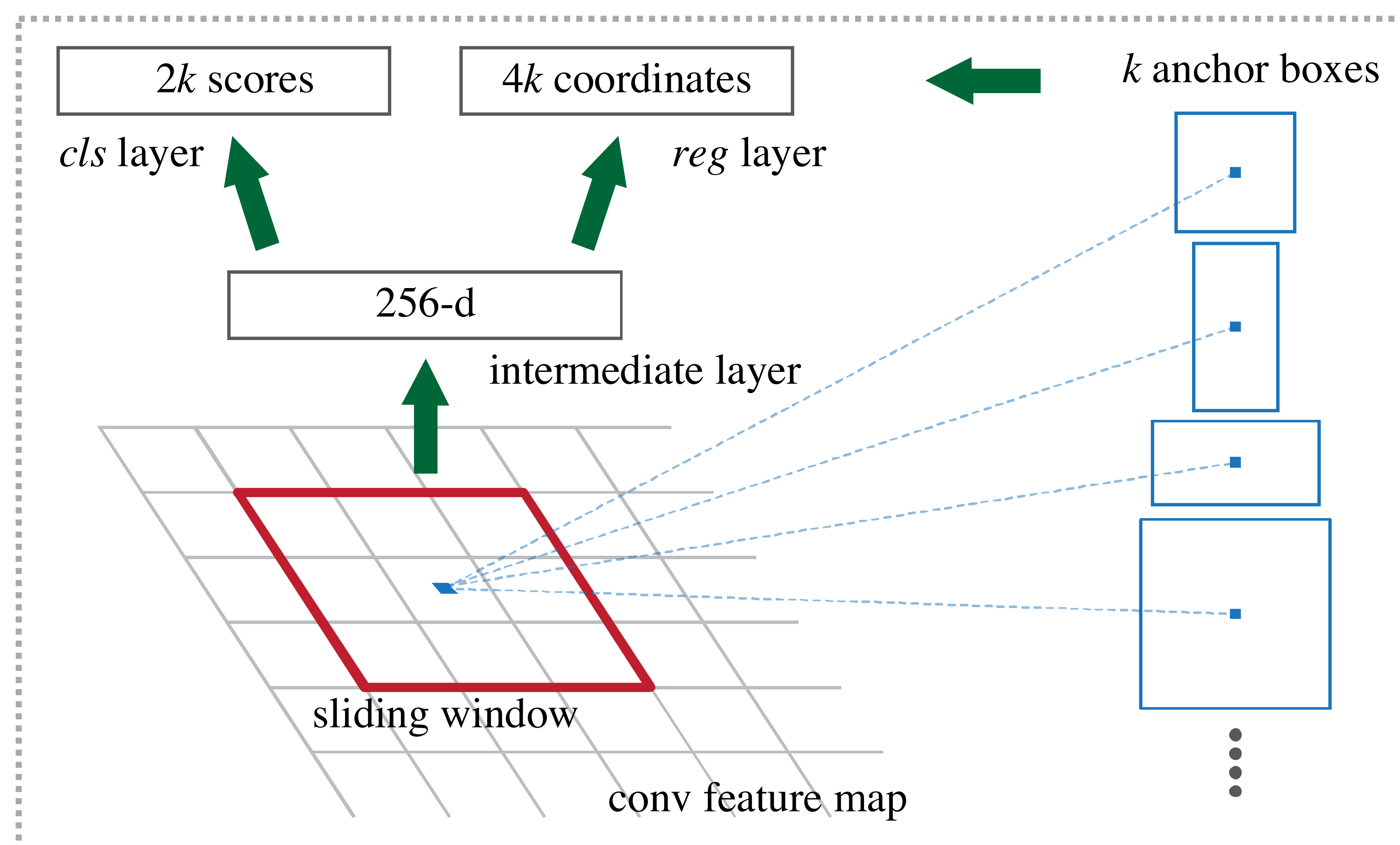
Faster R-CNN using a region proposal network (RPN)



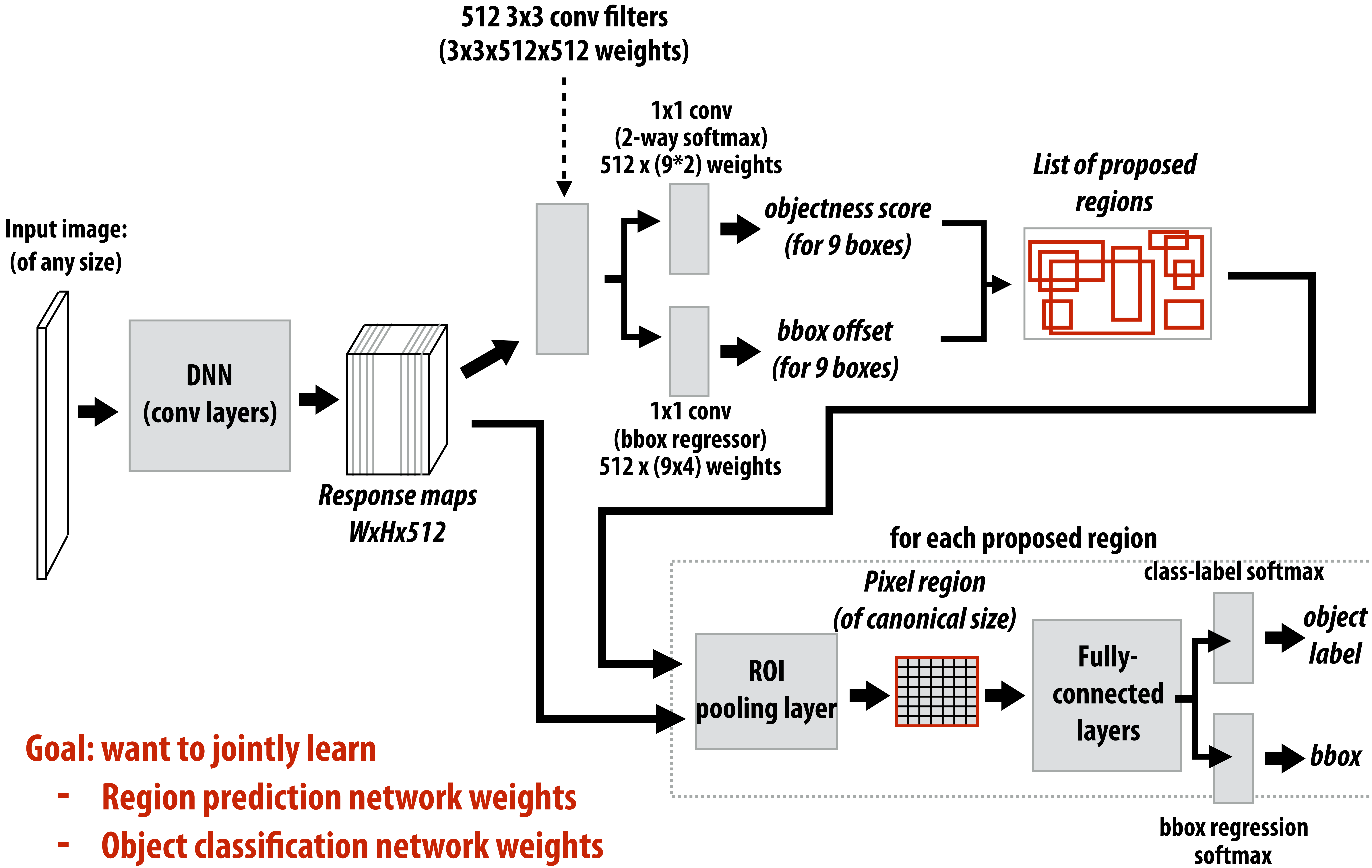
3x3 conv projects into 512-element vector per spatial position (assuming VGG input conv layers, receptive field for each output is ~228x228 pixels)

At each point assume 9 “anchor boxes” of various aspect ratios and scales

Given 512-element vector predict “objectness score” of each anchor + bbox correction to anchor



Training faster R-CNN



Goal: want to jointly learn
- Region prediction network weights
- Object classification network weights
- While constraining initial conv layers to be the same (for efficiency)

Alternating training strategy

- Train RPN
- Then use trained RPN to train Fast R-CNN
- Use conv layers from R-CNN to initialize RPN
- Fine-tune RPN
- Use updated RPN to fine tune Fast R-CNN
- Repeat…

- Notice: solution learns to predict boxes that are "good for object-detection task"
  - "End-to-end" optimization for object-detection task
  - Compare to using off-the-shelf object-proposal algorithm

Faster R-CNN results

Specializing region proposals for object-detection task yields better accuracy.

SS = selective search

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**

Shared convolutions improve algorithm performance:

Times in ms

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**

Summary

- Detailed knowledge of algorithm and properties of DNN used to gain algorithmic speedups
  - Not just "tune the schedule of the loops"

- Key insight: sharing results of convolutional layer computations:
  - Between different proposed regions
  - Between region proposal logic and detection logic

- Push for "end-to-end" training
  - Clean: back-propagate through entire algorithm to train all components at once
  - Better accuracy: globally optimize the various parts of the algorithm to be optimal for task (here: how to propose boxes learned simultaneously with detection logic)
  - Can constrain learning to preserve performance characteristics (conv layer weights must age shared across RPN and detection task)

Emerging theme

(from today's lecture and the Inception, SqueezeNet, and related readings)

- **Computer vision practitioners are "programming" via low-level manipulation of DNN topology**
  - **See shift from reasoning about individual layers to writing up of basic "microarchitecture" modules (e.g., Inception module)**

- **What programming model constructs or "automated compilation" tools could help raise the level of abstraction?**