CAP 5415: Computer Vision

Object Detection with Deep learning

Dr. Sedat Ozer

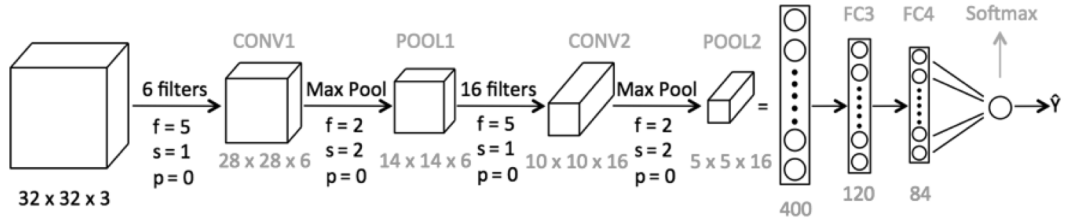
Announcements

- Using Email vs Webcourses
- Using GPUs
- **Midterm exam** is next week. Do you have questions?

Last lecture:

- LeNet-5
- VGG16
- ResNet

(Slightly Modified) LeNet-5



For: Grayscale image (single channel)

	Activation shape	Activation Size	# parameters
Input:	(32,32,**1**)	1,024	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	150 + 6 bias = **156**
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	(5x5x6) x 16= 2400 2400 + 16 = **2416**
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,000 + 120 = **48,120**
FC4	(84,1)	84	10,080 + 84 = **10,164**
Softmax	(10,1)	10	84x10 + 10 = **850**

For: Colored (RGB) image

	Activation shape	Activation Size	# parameters
Input:	(32,32,**3**)	3,072	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	450 + 6 bias = **456**
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	(5x5x6) x 16= 2400 2400 + 16 = **2416**
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,000 + 120 = **48,120**
FC4	(84,1)	84	10,080 + 84 = **10,164**
Softmax	(10,1)	10	84x10 + 10 = **850**

Tensorflow output for the (grayscale) model summarizing the model parameters:

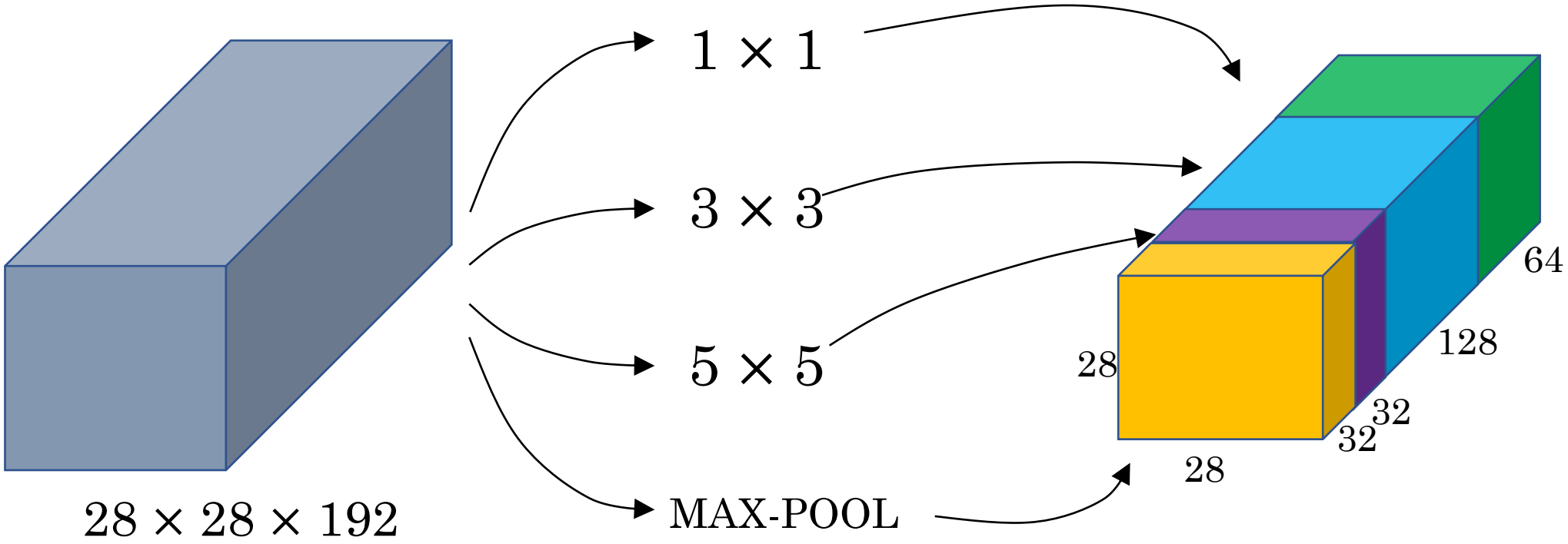
```
Variables: name (type shape) [size]
---------
Variable:0 (float32_ref 5x5x1x6) [150, bytes: 600]
conv1_biases:0 (float32_ref 6) [6, bytes: 24]
Variable_1:0 (float32_ref 5x5x6x16) [2400, bytes: 9600]
conv2_biases:0 (float32_ref 16) [16, bytes: 64]
Variable_2:0 (float32_ref 400x120) [48000, bytes: 192000]
fc1_biases:0 (float32_ref 120) [120, bytes: 480]
Variable_3:0 (float32_ref 120x84) [10080, bytes: 40320]
fc2_biases:0 (float32_ref 84) [84, bytes: 336]
Variable_4:0 (float32_ref 84x10) [840, bytes: 3360]
fc3_biases:0 (float32_ref 10) [10, bytes: 40]
Total size of variables: 61706
Total bytes of variables: 246824
sedat@sedat-ThinkPad-P50:~/
```

Inception Network

(GoogleNet)

http://knowyourmeme.com/memes/we-need-to-go-deeper

Main motivation for inception network



[Szegedy et al. 2014. Going deeper with convolutions]

What does a 1 × 1 convolution do?

Input is 2D (one channel)

3	1	3	6	5	8
3	2	4	1	3	4
2	1	3	4	9	2
4	5	8	1	7	9
1	5	3	3	4	3
5	4	4	8	3	1

6 × 6

*

2

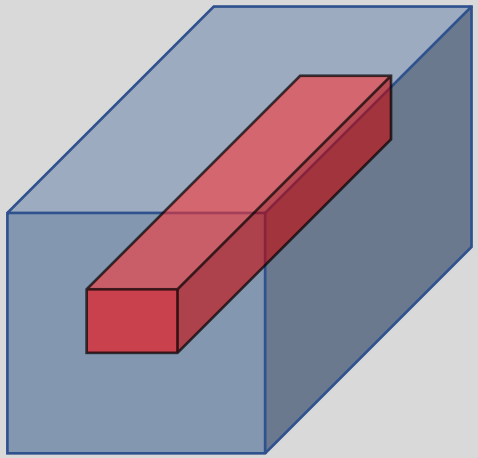
1 × 1

=

6	2	6	.	.	.
6	4	8	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

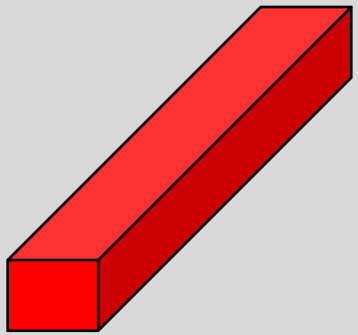
6 × 6

Input is 3D Volume (multiple channels)



6 × 6 × 32

*



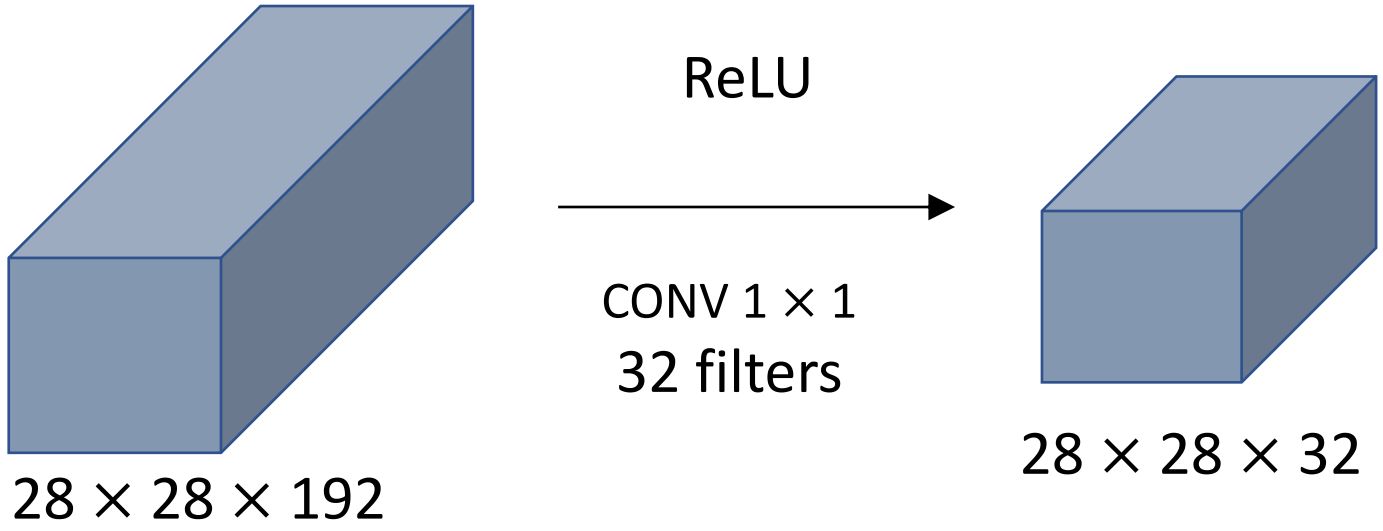
1 × 1 × 32

=

6 × 6 (x 1)

[Lin et al., 2013. Network in network]

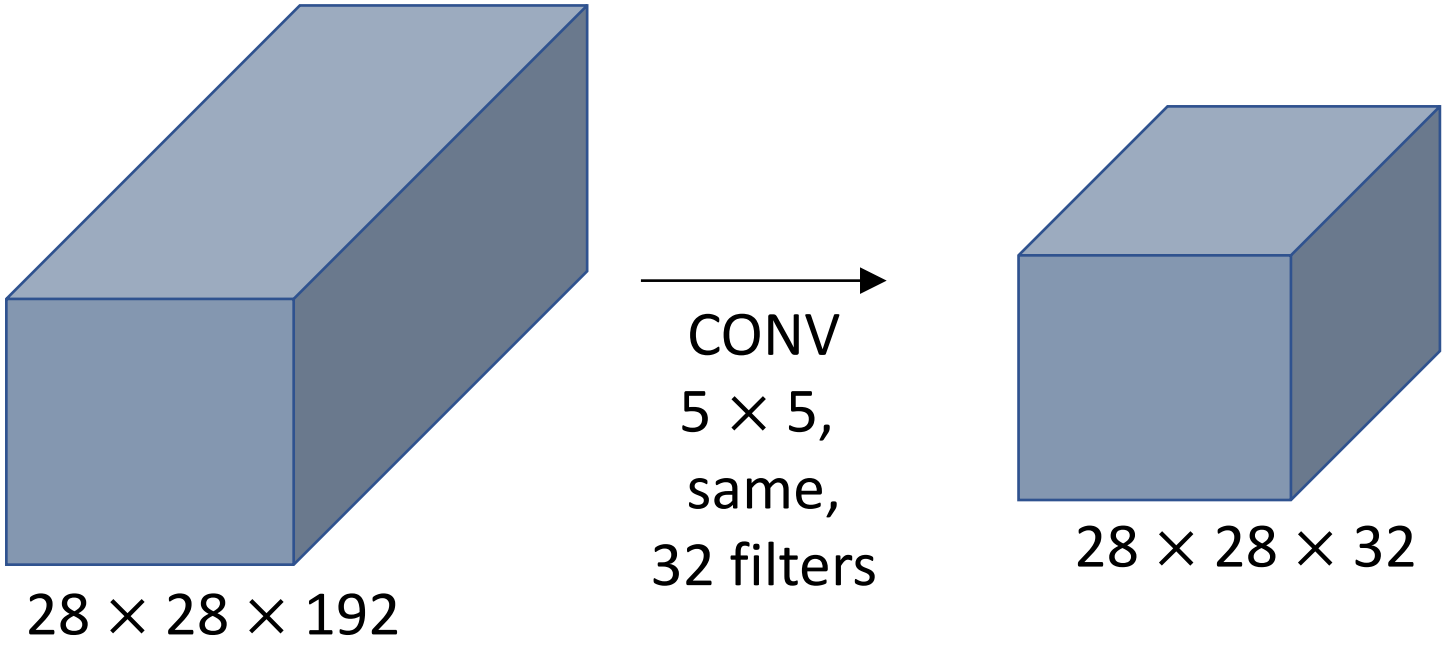
1×1 convolutions can shrink the layer dims



Helps reducing/resizing the number of parameters!

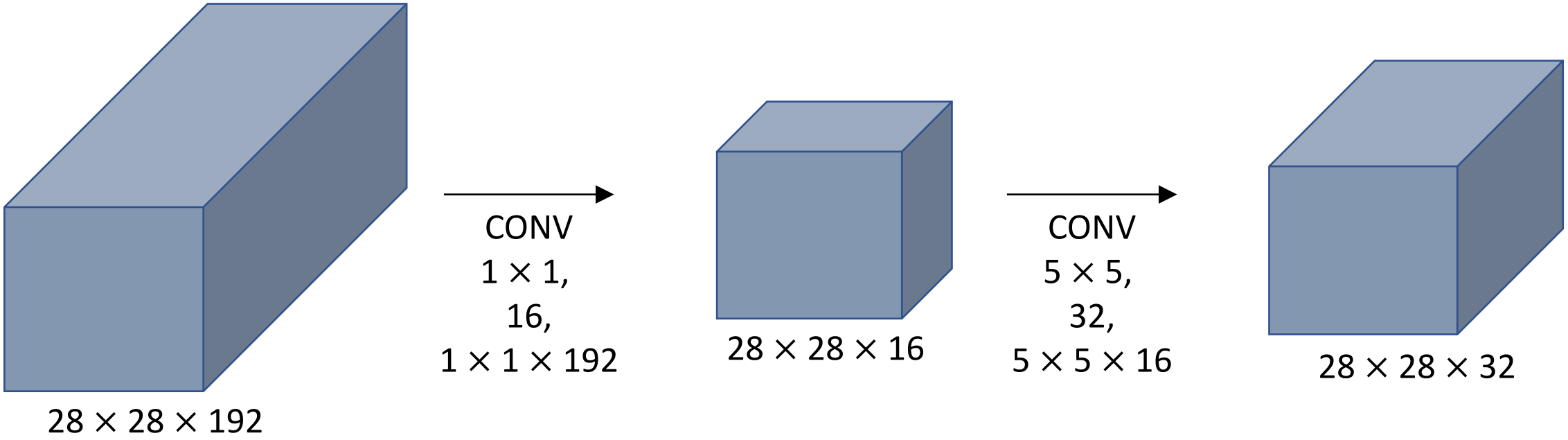
[Lin et al., 2013. Network in network]

The problem of computational cost



- One filter volume is: 5x5x192 for one voxel in the output volume (in the activation map).
- Since we have 28x28x32 dimensional volume at the output, we need:
  (5x5x192) x (28x28x32) multiplications! (~120Million)

Using 1×1 convolution for reducing the computation

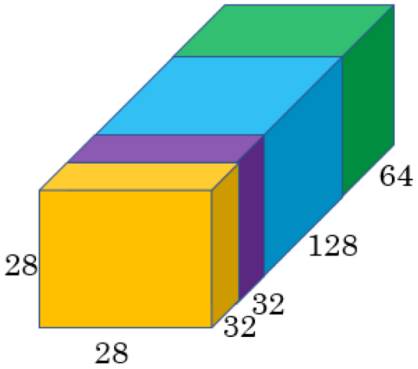


Multiplications needed: (28x28x16) x (1x1x192) = ~ 2.4 Million (28x28x32) x (5x5x16) = ~ 10 Million

Total: 12.4 Million << 120Million

Inception Net (GooLeNet) incorporates this idea!

Inception module



Previous Activation

1 × 1 CONV

1 × 1 CONV

3 × 3 CONV

1 × 1 CONV

5 × 5 CONV

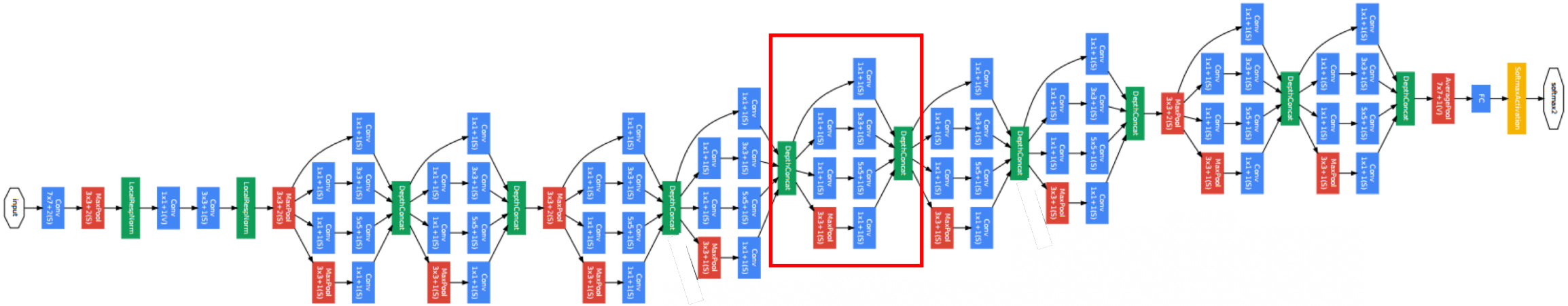
MAXPOOL 3 × 3,s = 1 same

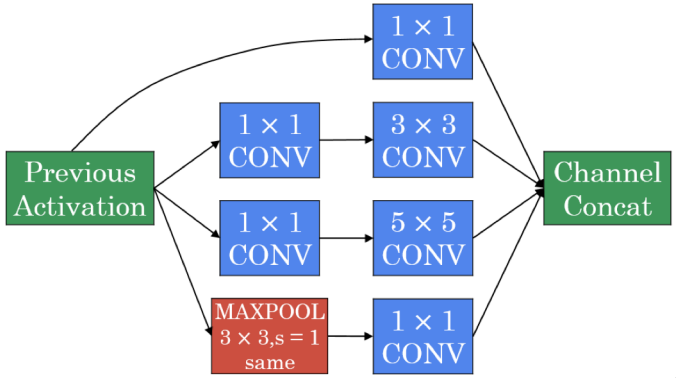
1 × 1 CONV

Channel Concat

Inception Network (Google)

GooLeNet or GoogLeNet





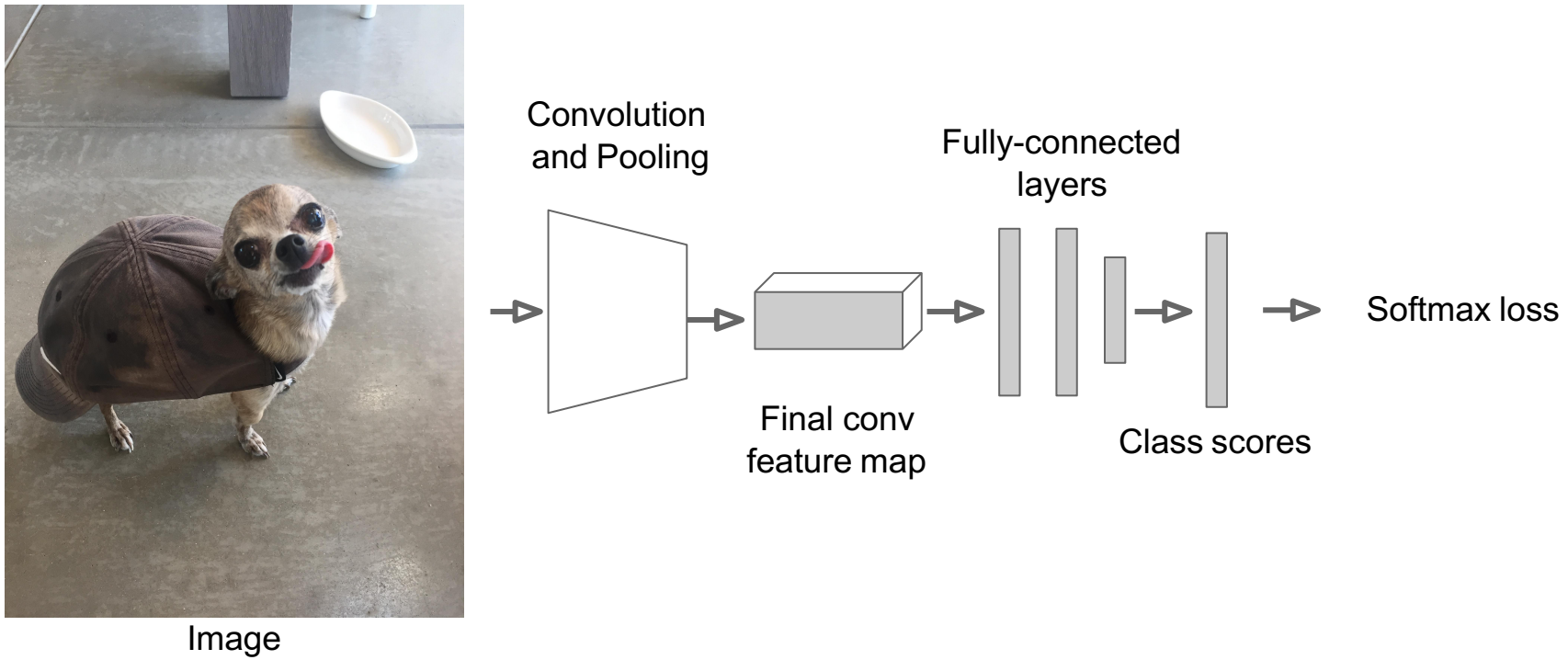
Local Response Normalization (LRN)

[Szegedy et al., 2014, Going Deeper with Convolutions]

Classification + Localization

Simple Recipe for Classification + Localization

Step 1: Train (or download) a classification model (AlexNet, VGG, GoogLeNet)



Problem: Is there a **turtle** in this picture? If yes, localize!

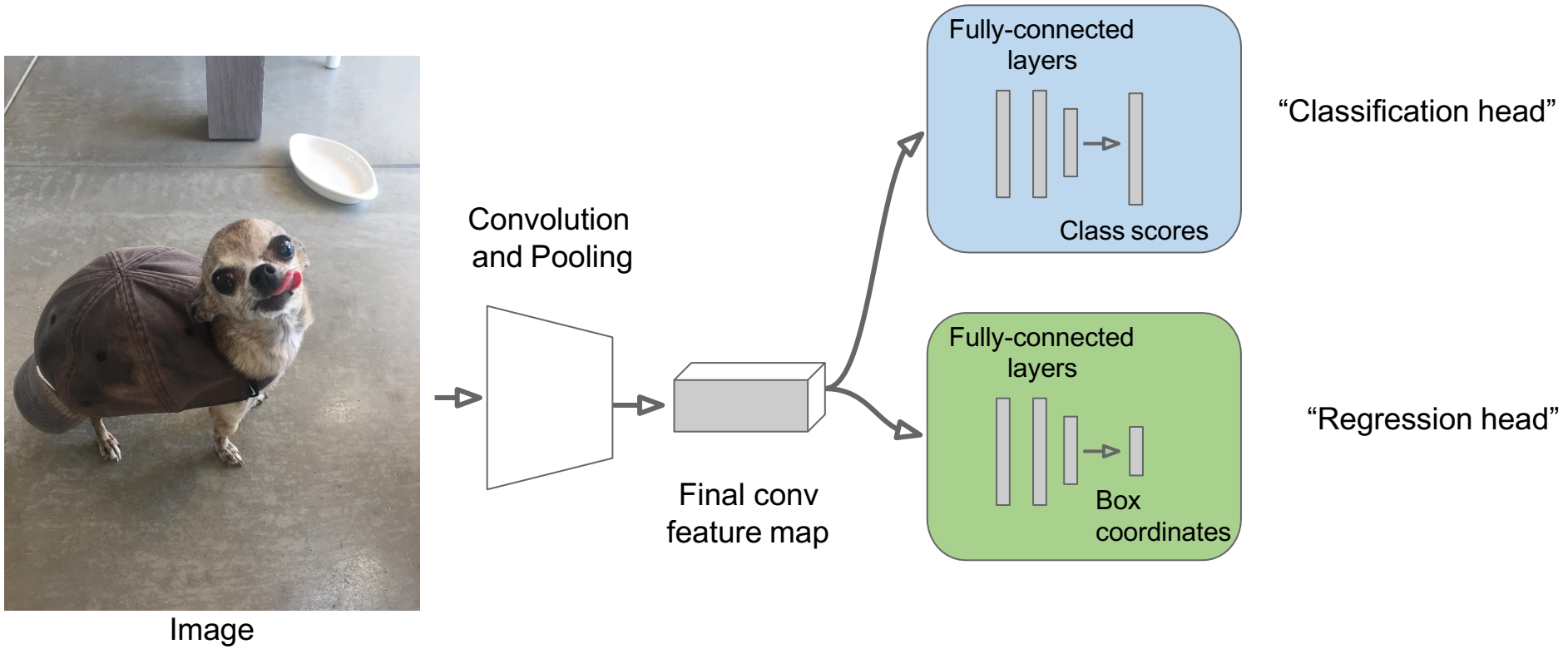
Questions:

- What is the purpose of the detection task?
- What do we compute for detection at the output? (Does it differ from classification? If so, how?)

Simple Recipe for Classification + L

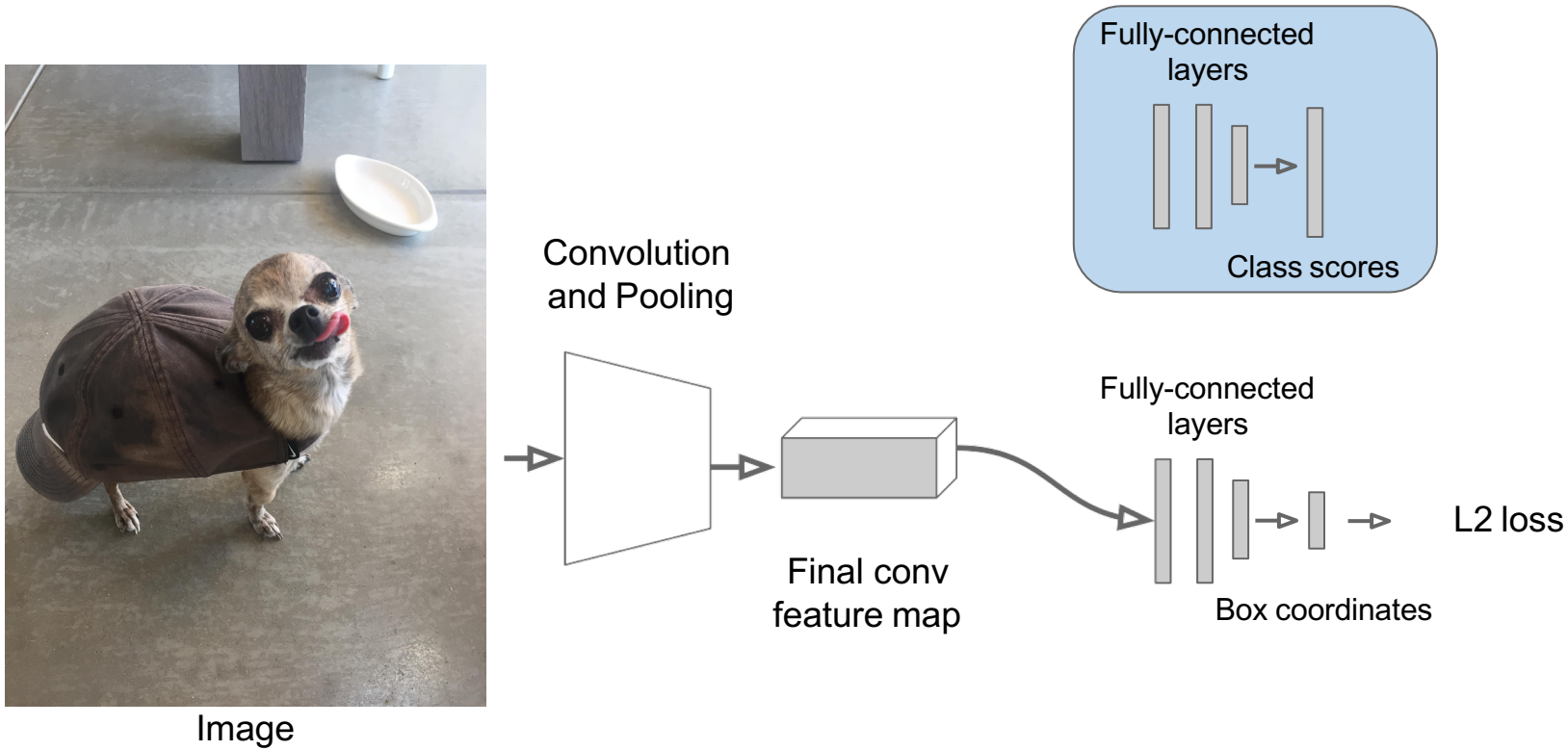
Question: What is regression again?

Step 2: Attach new fully-connected "regression head" to the network.



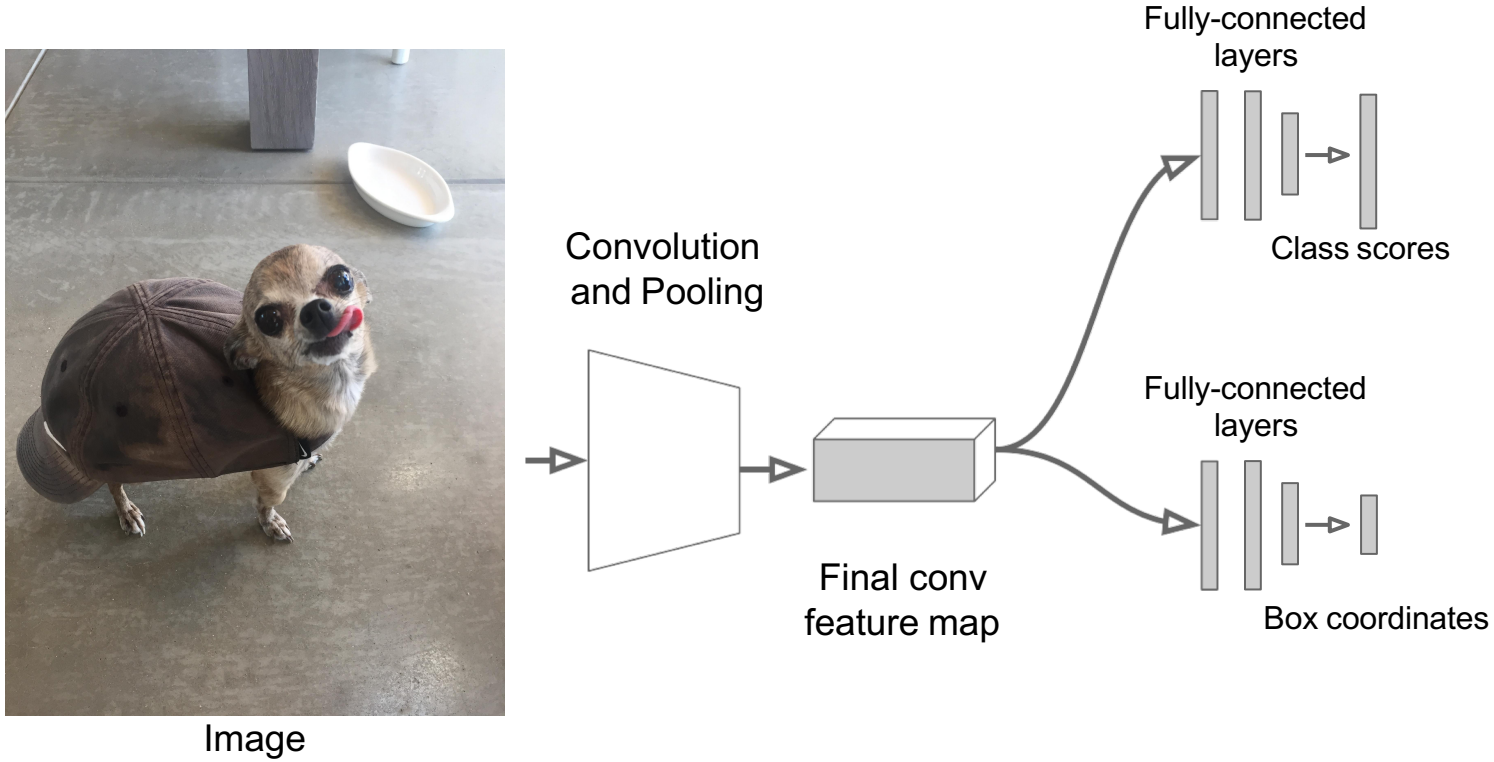
Simple Recipe for Classification + Localization

Step 3: Train the regression head only with (mini) batch Gradient Descent (or SGD) and L2 loss



Simple Recipe for Classification + Localization

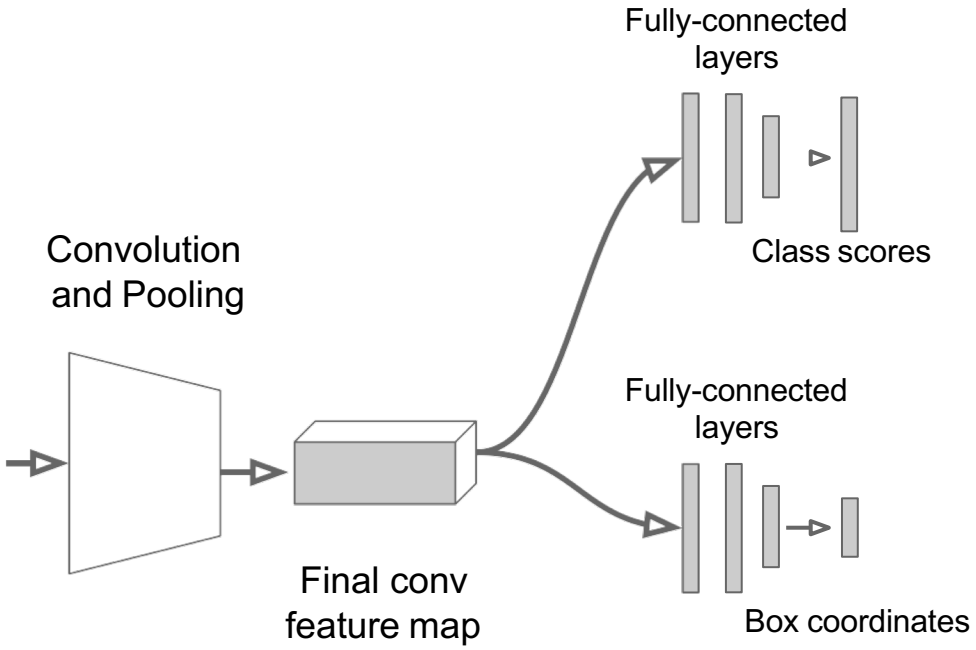
Step 4: At test time use both heads.



“Per-class" or “class agnostic" regression

Assume classification over C classes:

Image



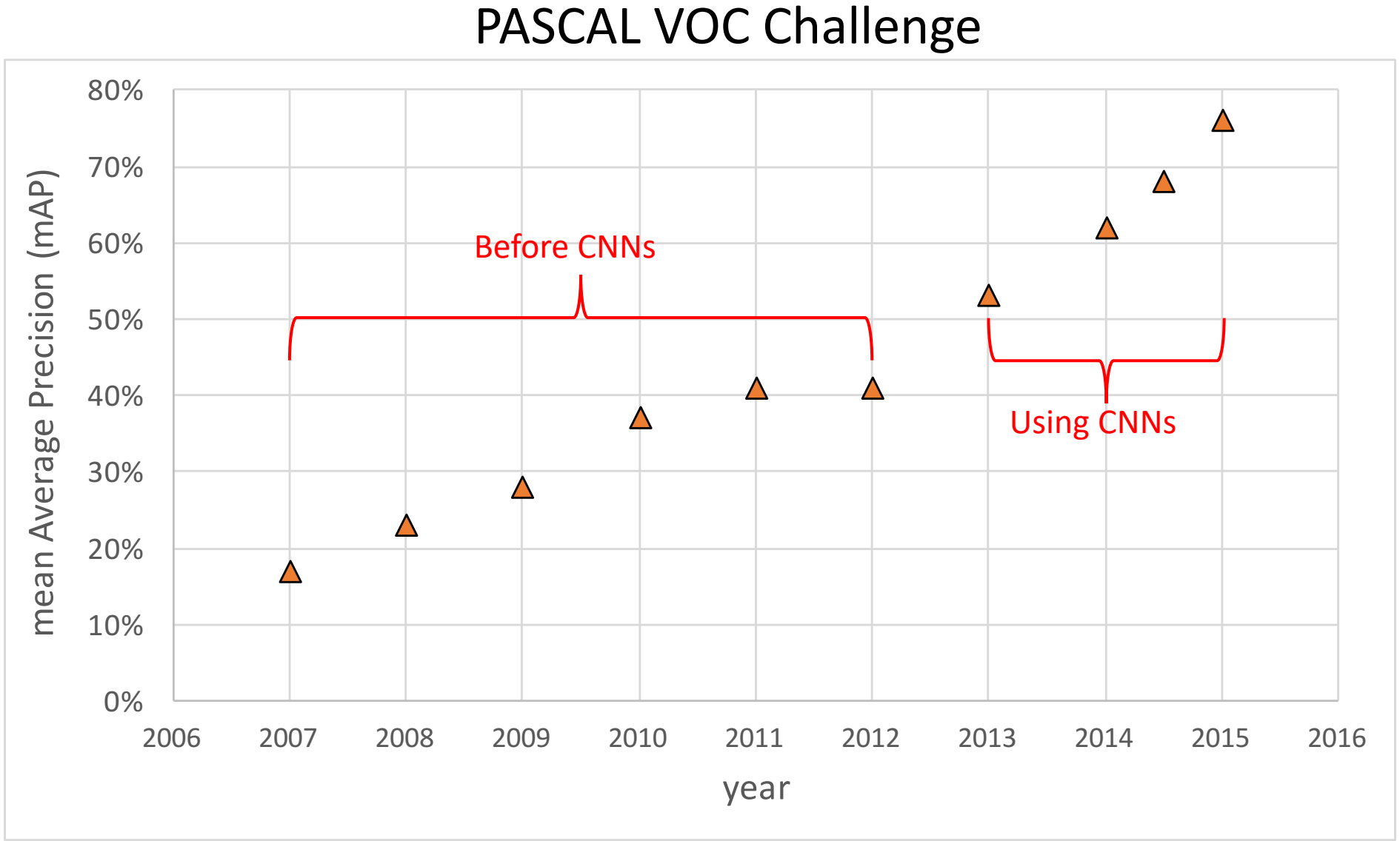
Classification head:
C numbers
(one per class)

Class agnostic:
4 numbers
(one box)

Class specific:
C x 4 numbers
(one box per class)

Detection

Object detection progress over years



Deformable Parts Models (before the CNN era)

Source: R. Girshick

Detection as Regression?

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

= 16 numbers

Detection as Regression?

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

= 8 numbers

Detection as Regression?

CAT, (x, y, w, h)
CAT, (x, y, w, h)
....
CAT (x, y, w, h)

= many numbers

Need variable sized outputs

Detection as Classification

CAT? NO

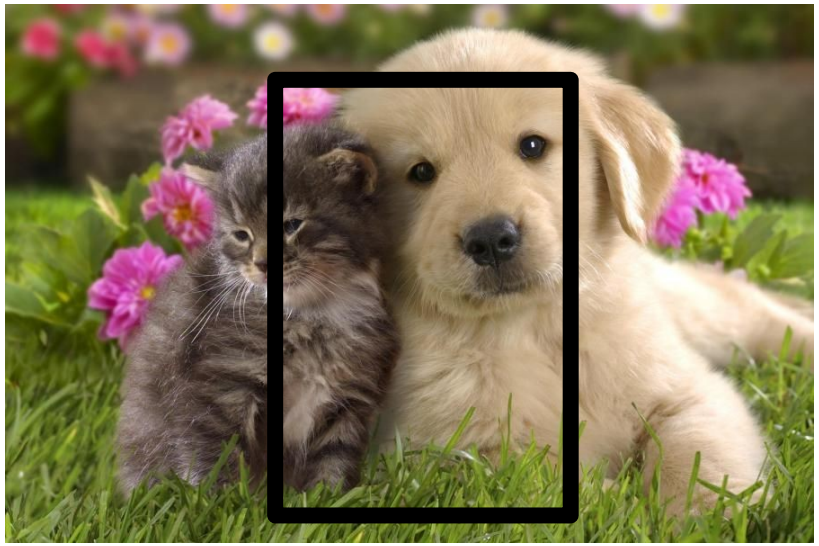
DOG? NO

Detection as Classification

CAT? YES!

DOG? NO

Detection as Classification



CAT? NO

DOG? NO

Detection as Classification

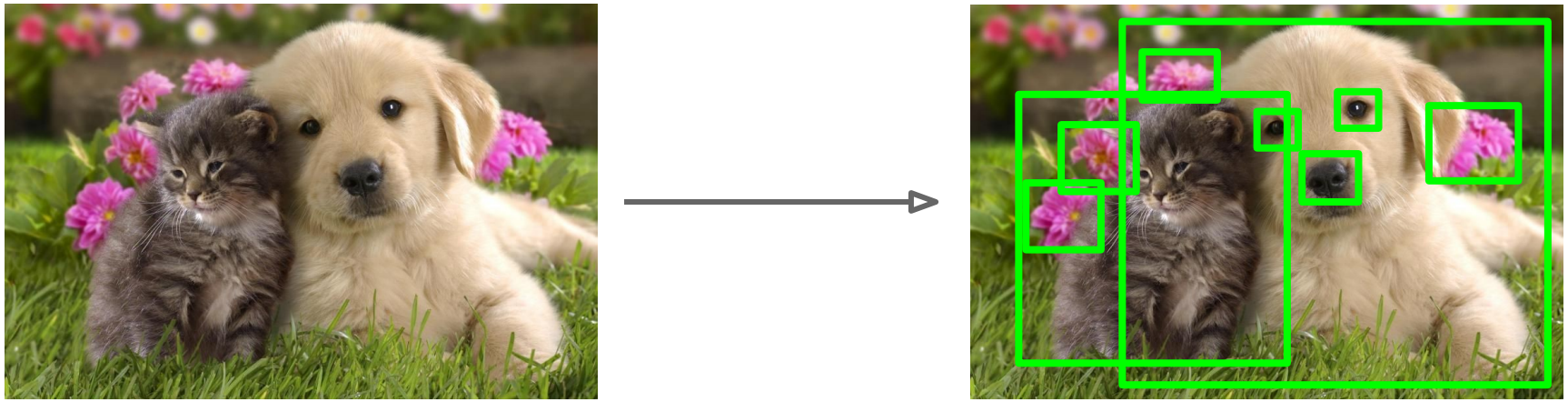
Problem: Need to test many positions and scales, and use a computationally demanding classifier (CNN)

- Search at different scales
- Search at different positions

Solution: Only look at a tiny subset of possible positions

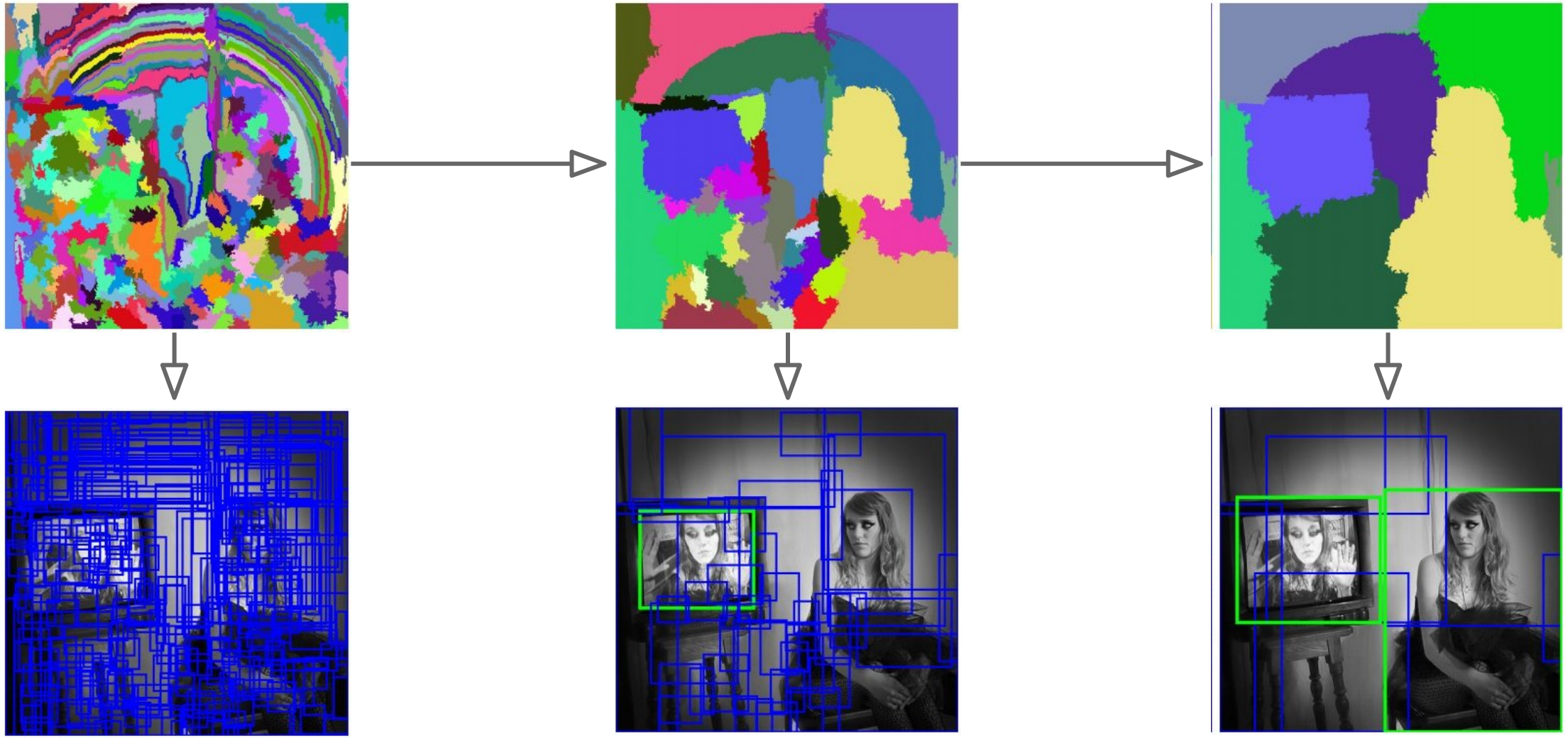
Region Proposals

- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



Region Proposals: Selective Search

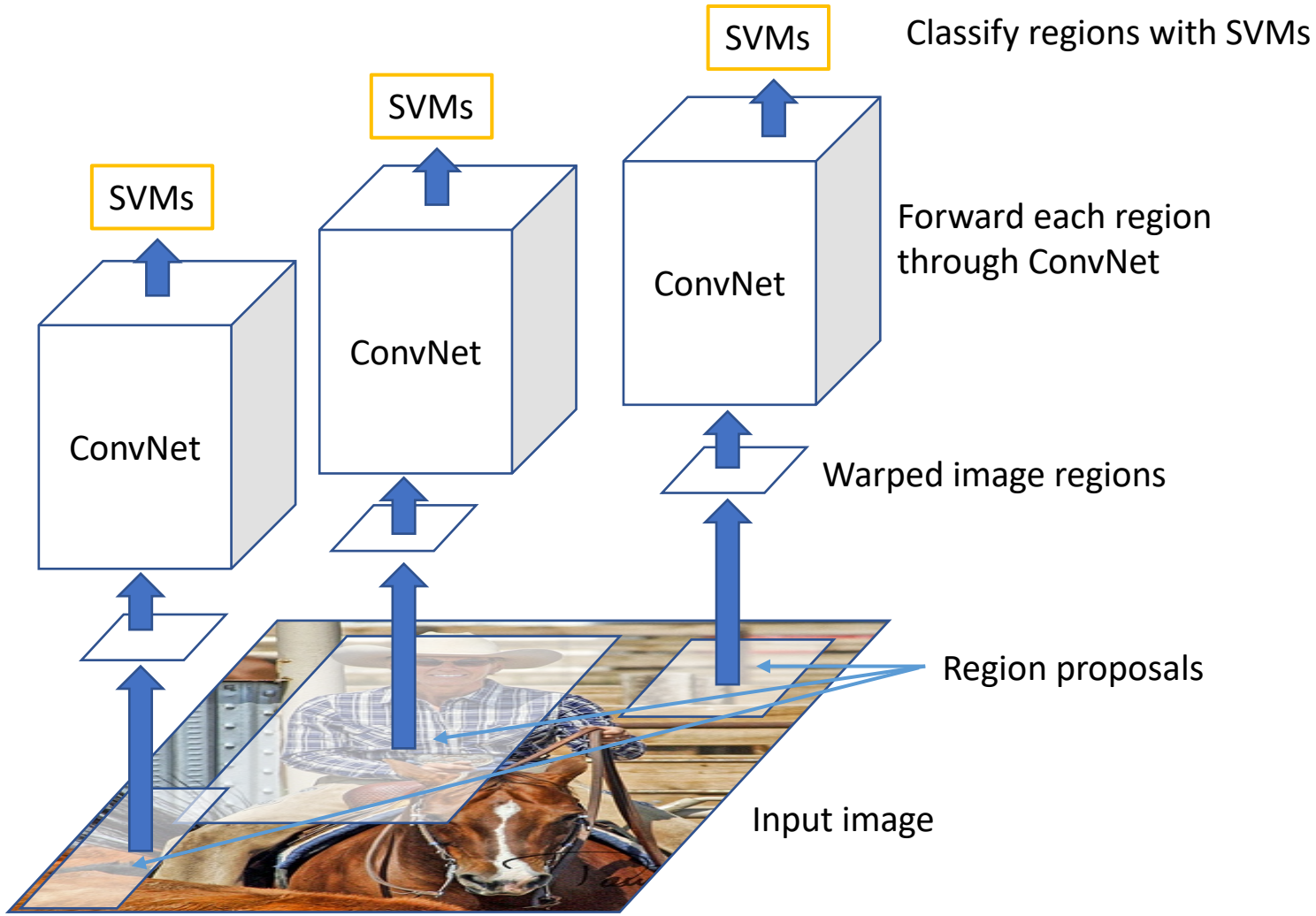
Bottom-up segmentation, merging regions at multiple scales



Convert regions to boxes

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

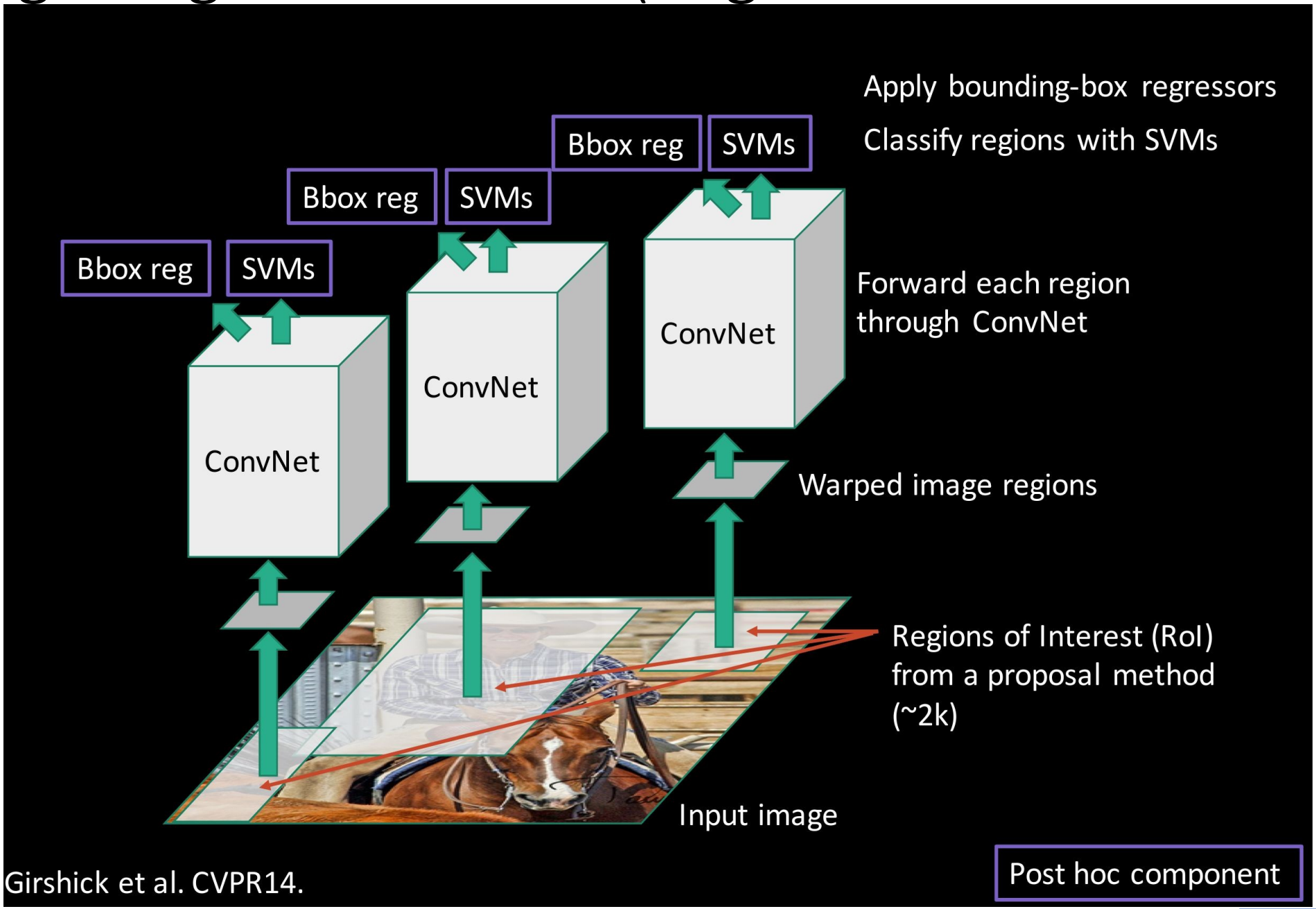
R-CNN: Region proposals + CNN features (Regions with CNN features)



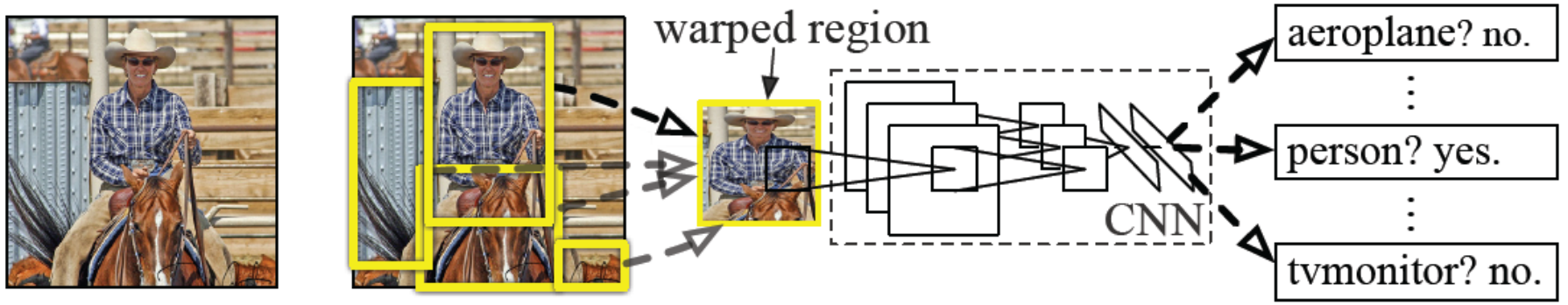
Source: R. Girshick

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

Putting it together: R-CNN (Regions with CNN features)



R-CNN details



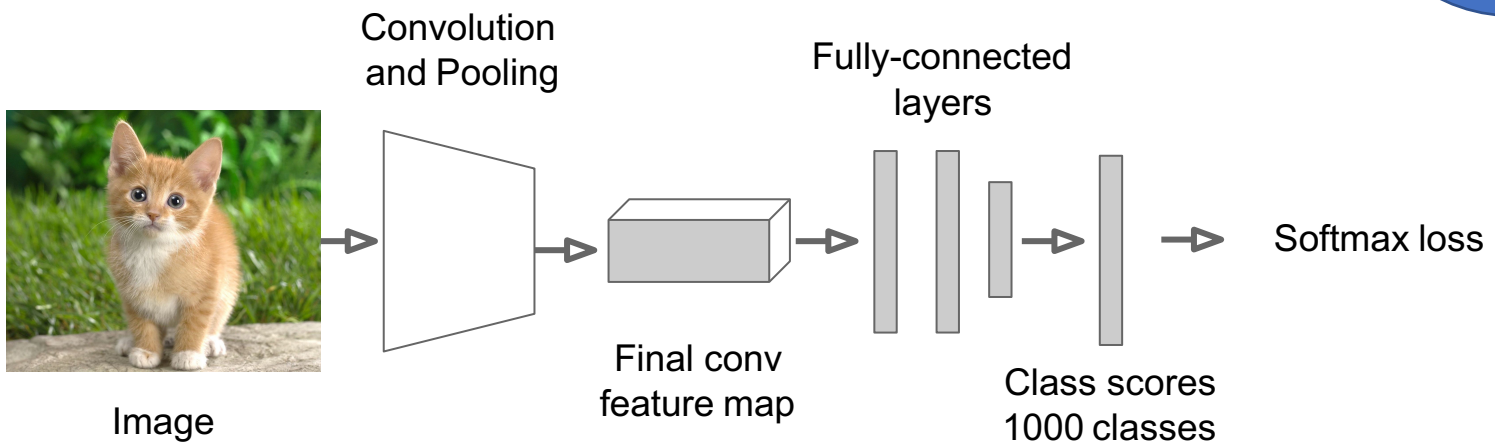
- **Regions**: uses ~2000 Selective Search proposals
- **Network**: uses AlexNet *pre-trained* on ImageNet (1000 classes), *fine-tuned* on PASCAL (21 classes)
- **Final detector**:
  - first warp proposal regions,
  - then extract fc7 network activations (4096 dimensions),
  - Finally, classify with linear SVM
- **Bounding box regression** is also used to refine box locations
- **Performance:** mAP of 53.7% on PASCAL 2010 dataset
  (Compare that value to: 35.1% for Selective Search and to: 33.4% for DPM).

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

R-CNN Training (initialization)

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

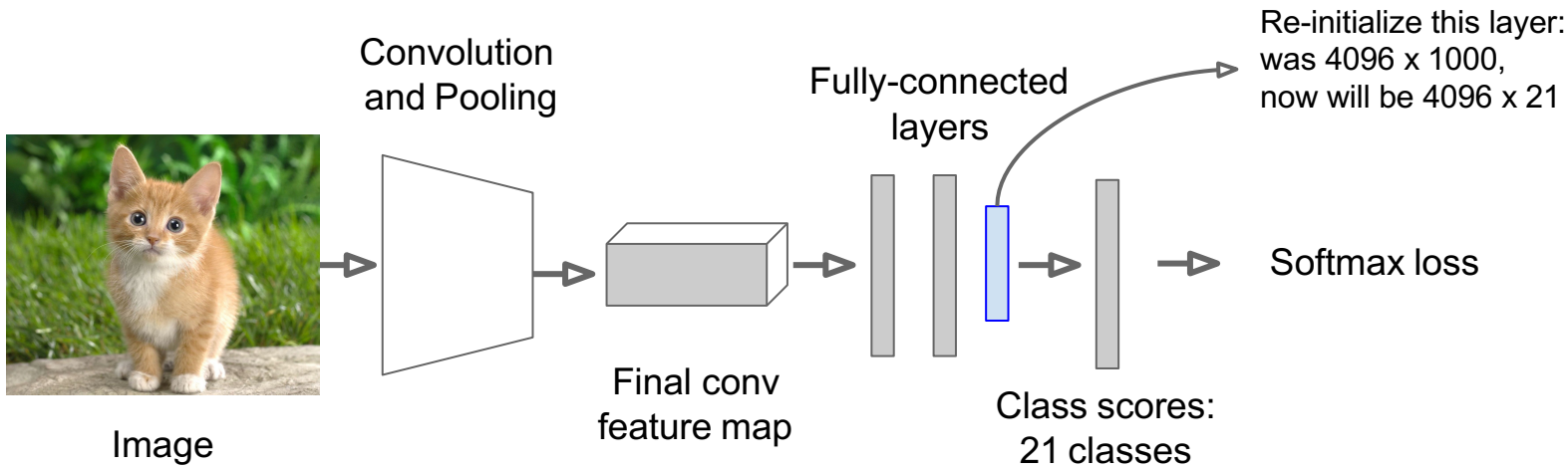
What is in a model?
What do we actually download?



R-CNN Training (Fine-tuning)

Step 2: **"Fine-tune"** model for detection

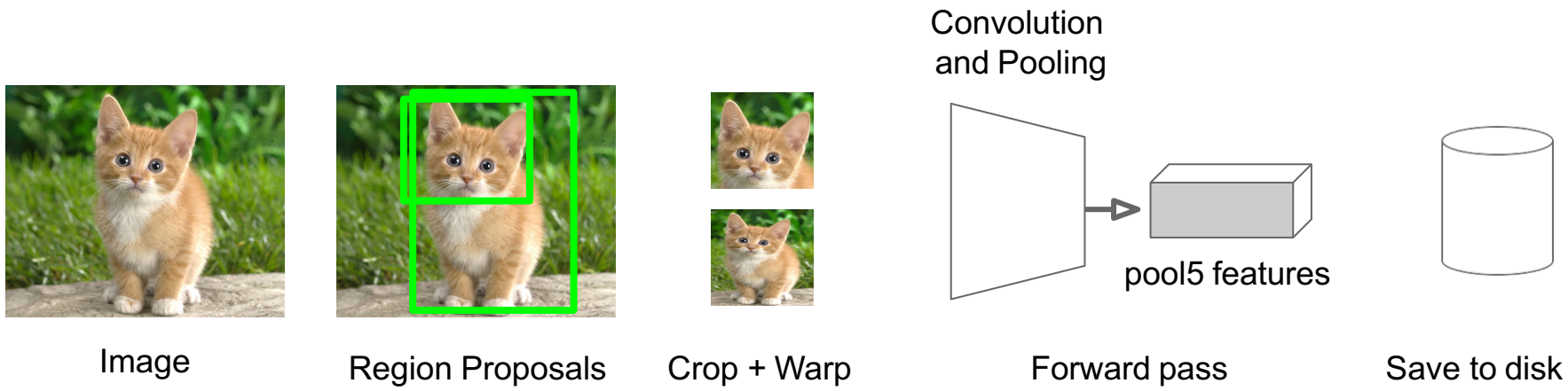
- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images



R-CNN Training (feature extraction)

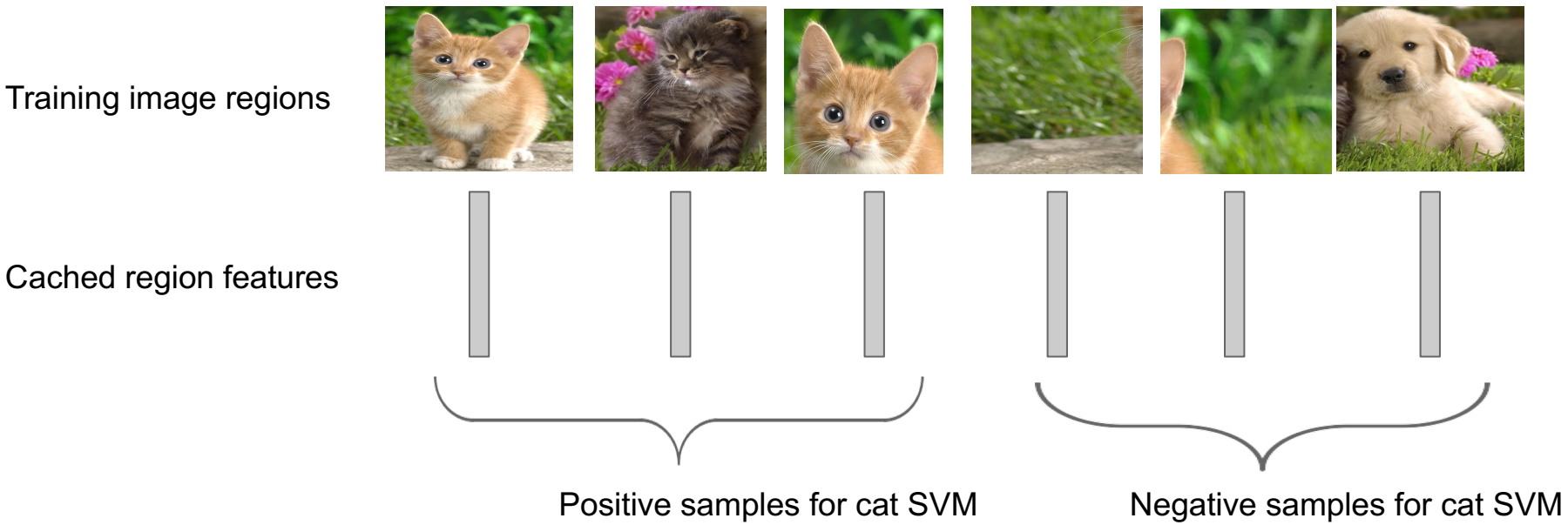
Step 3: Extract features

- 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!



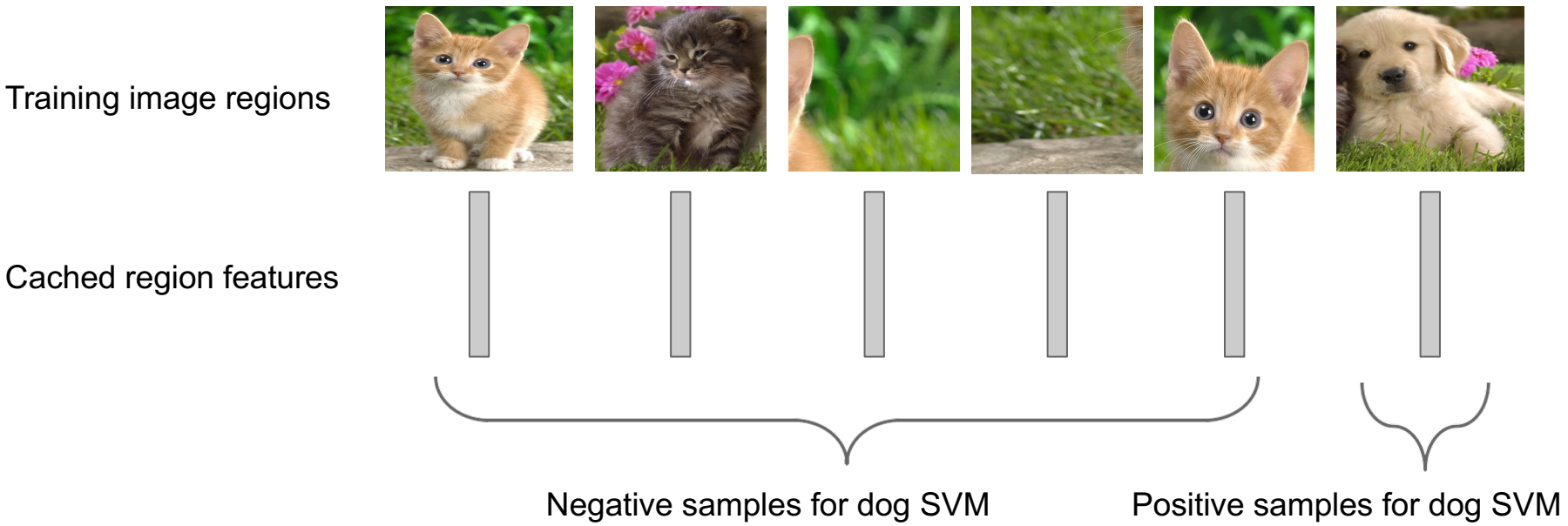
R-CNN Training (train classifier)

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



R-CNN Training (train classifier)

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



R-CNN Training (bounding box regression/prediction)

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

Training image regions:

Cached region features

Regression **targets:**
(dx, dy, dw, dh)
Normalized coordinates

(0, 0, 0, 0)
Proposal is good

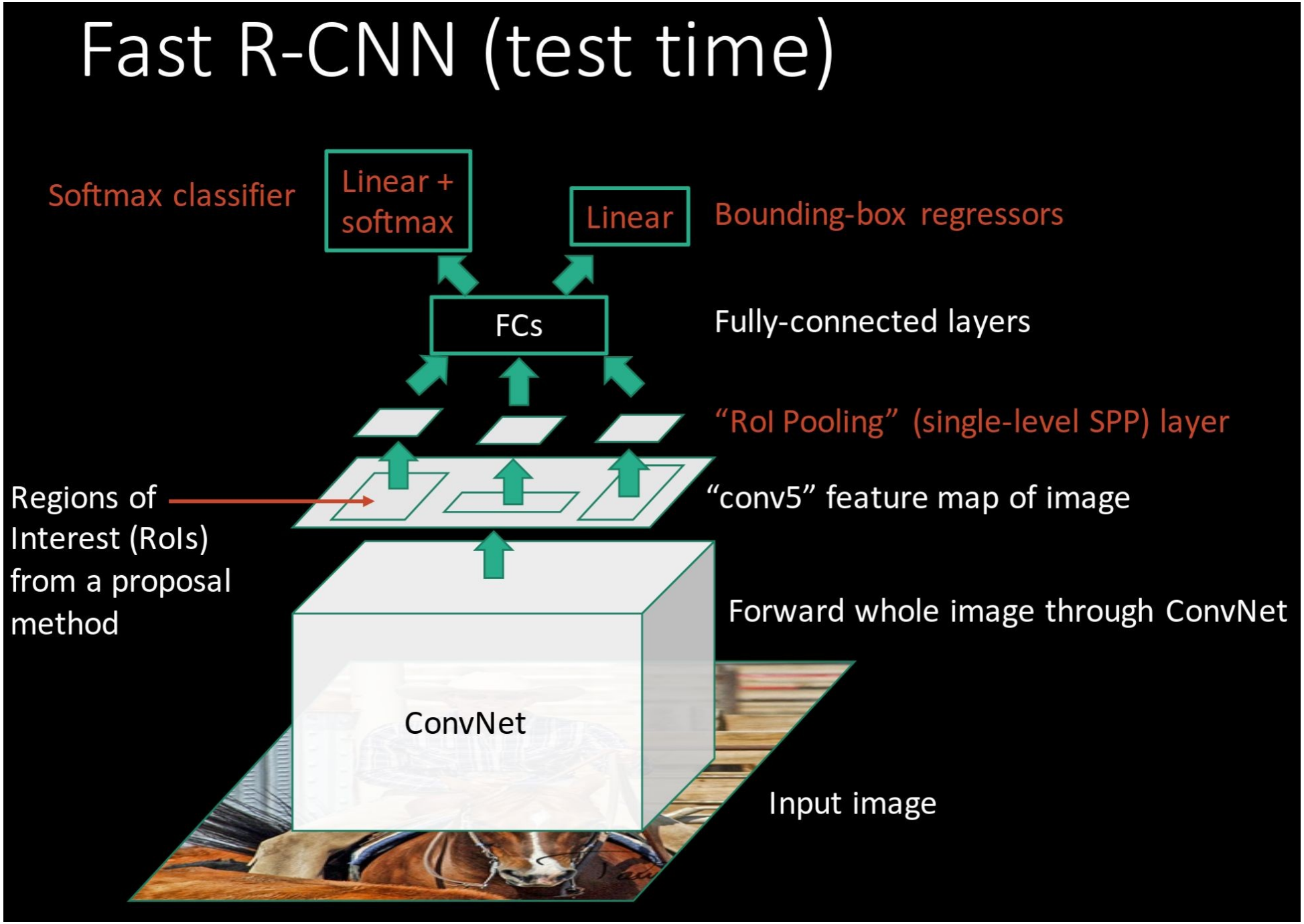
(.25, 0, 0, 0)
Proposal too far to left

(0, 0, -0.125, 0)
Proposal too wide

Object Detection: Datasets

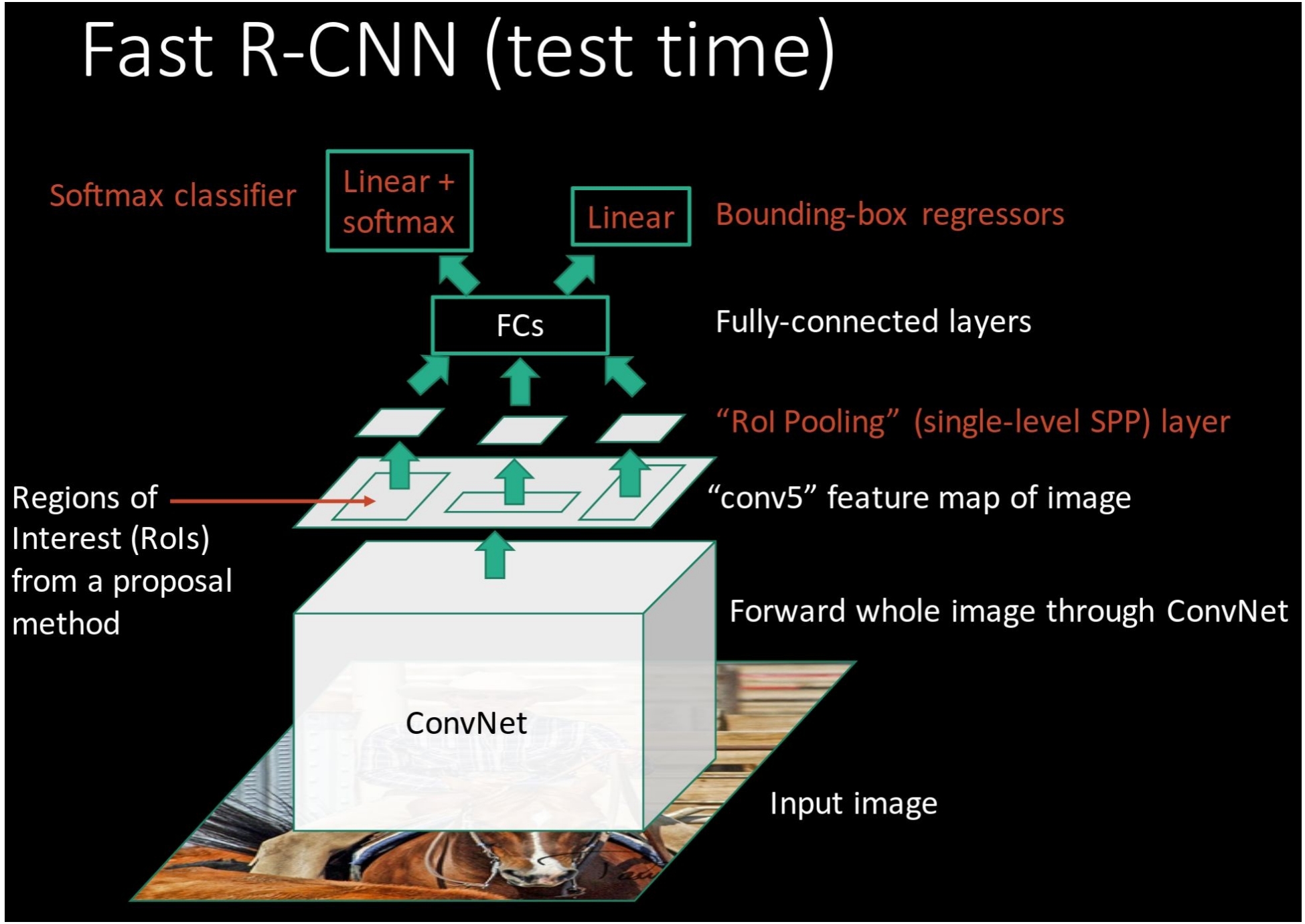
	PASCAL VOC (2010)	ImageNet Detection (ILSVRC 2014)	MS-COCO (2014)
Number of classes	20	**200**	80
Number of images (train + val)	~20k	**~470k**	~120k
Mean objects per image	2.4	1.1	**7.2**

Fast R-CNN (test time)



Slide credit: Ross Girschick, Girschick, "Fast R-CNN", ICCV 2015

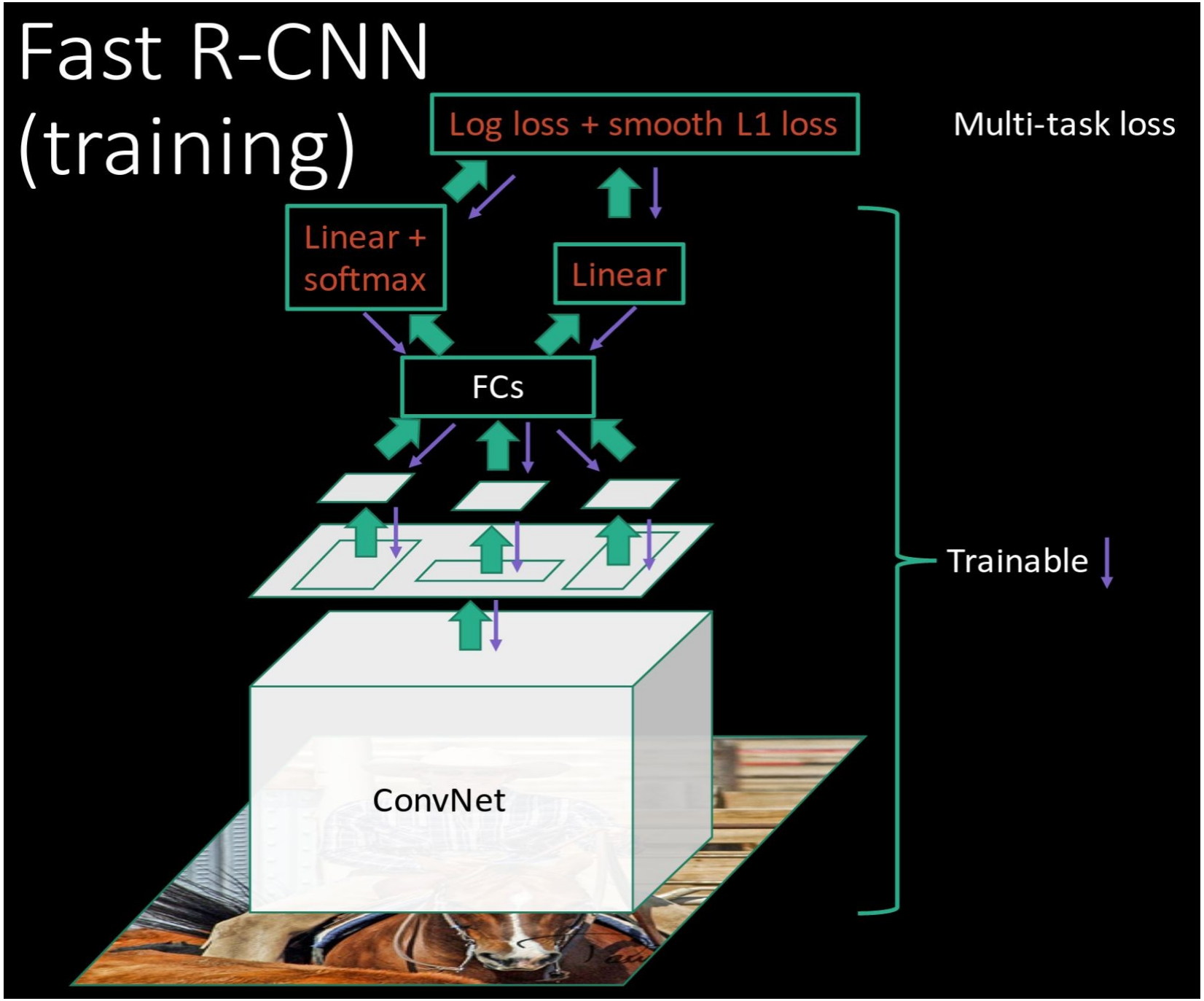
Fast R-CNN (test time)



R-CNN Problem #1:
Slow at test-time due to independent forward passes of the CNN

Solution:
Share computation of convolutional layers between proposals for an image

Slide credit: Ross Girschick, Girschick, "Fast R-CNN", ICCV 2015



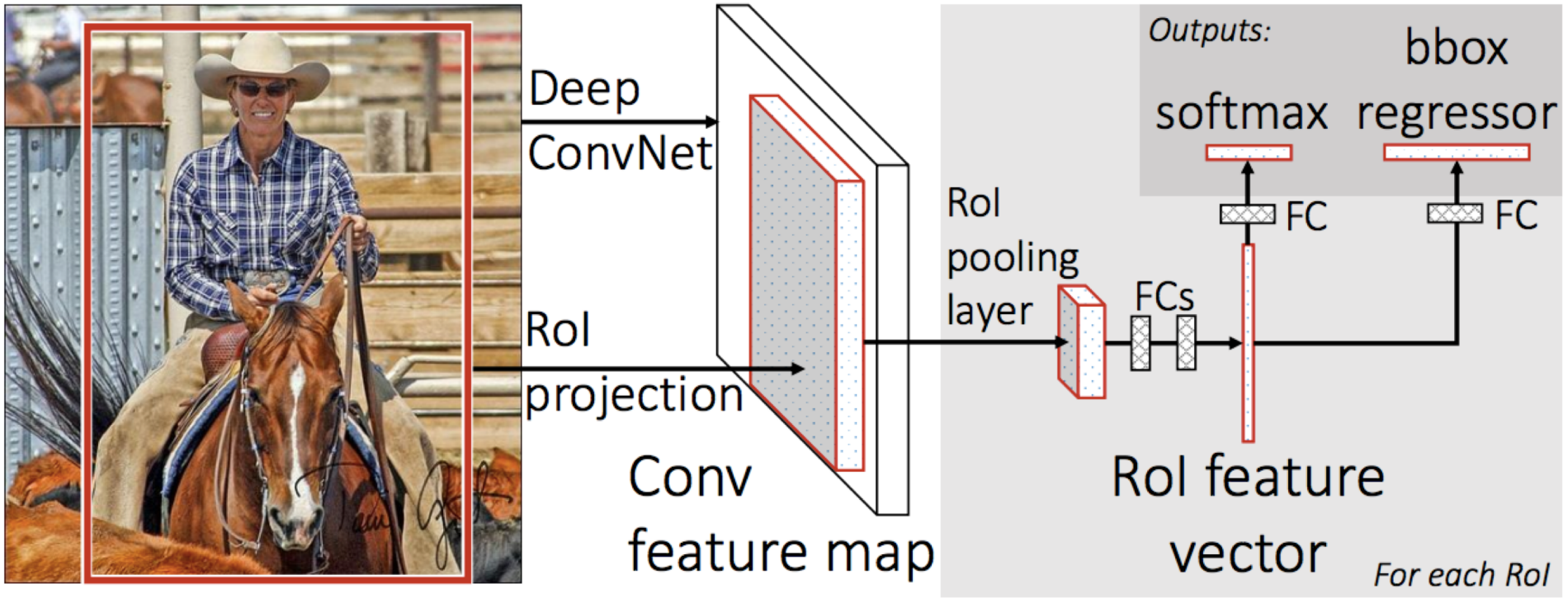
R-CNN Problem #2:
Post-hoc training: CNN not updated in response to final classifiers and regressors

R-CNN Problem #3:
Complex training pipeline

Solution:
Just train the whole system end-to-end all at once!

Slide credit: Ross Girschick

Fast R-CNN: Another view

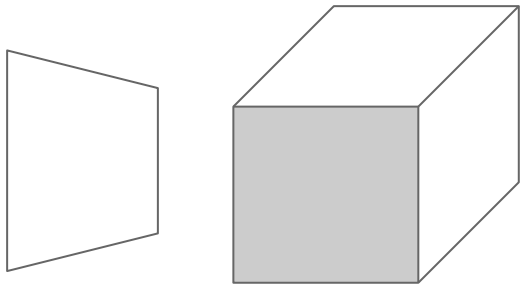


R. Girshick, Fast R-CNN, ICCV 2015

Fast R-CNN: Region of Interest Pooling

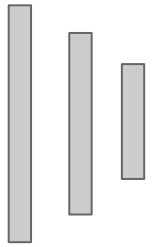
Hi-res input image:
3 x 800 x 600
with region
proposal

Convolution
and Pooling



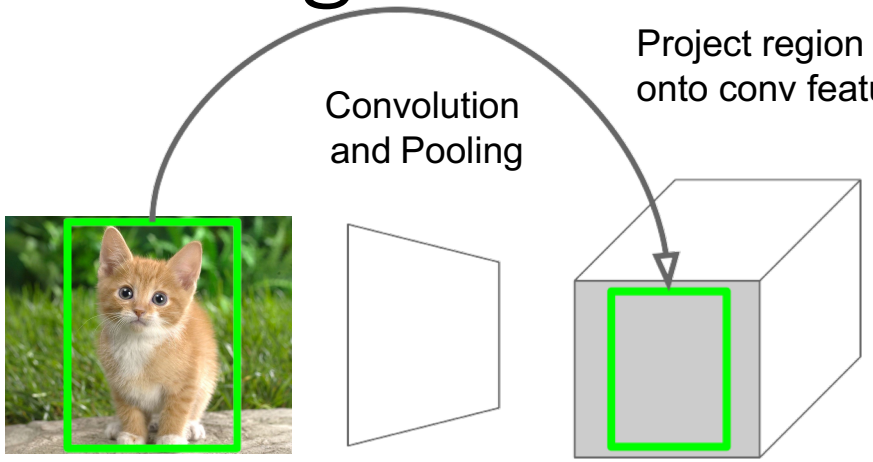
Hi-res conv features:
C x H x W
with region proposal

Fully-connected
layers



Problem: Fully-connected layers expect low-res conv features: C x h x w

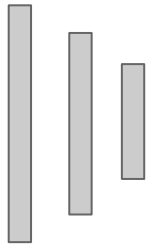
Fast R-CNN: Region of Interest Pooling



Hi-res input image:
3 x 800 x 600
with region
proposal

Hi-res conv features:
C x H x W
with region proposal

Fully-connected
layers

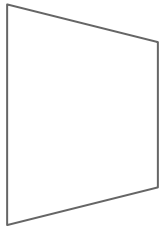


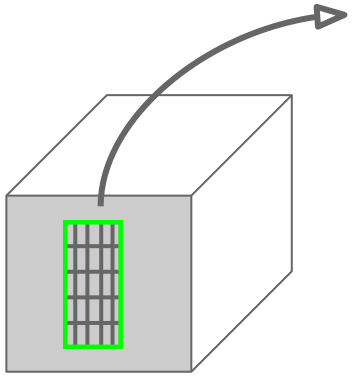
Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling

Hi-res input image:
3 x 800 x 600
with region
proposal

Convolution
and Pooling

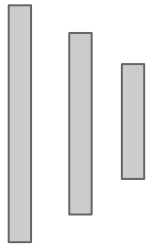




Hi-res conv features:
C x H x W
with region proposal

Divide projected
region into h x w grid

Fully-connected
layers

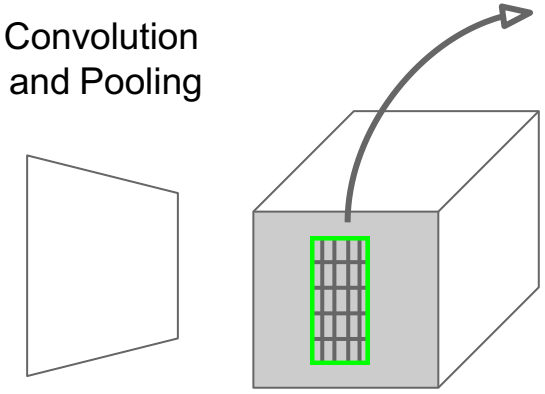


Problem: Fully-connected layers expect low-res conv features: C x h x w

Fast R-CNN: Region of Interest Pooling

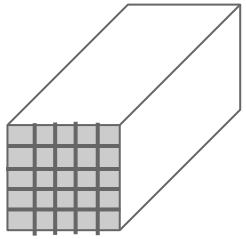
Hi-res input image:
3 x 800 x 600
with region
proposal

Convolution
and Pooling



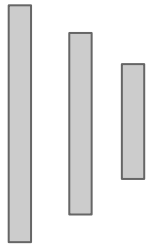
Hi-res conv features:
C x H x W
with region proposal

Max-pool within
each grid cell



RoI conv features:
C x h x w
for region proposal

Fully-connected
layers

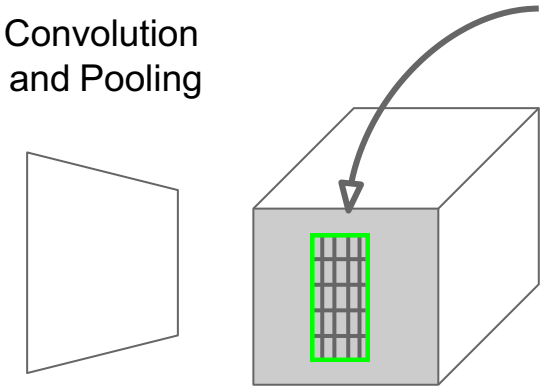


Fully-connected layers expect
low-res conv features:
C x h x w

Fast R-CNN: Region of Interest Pooling

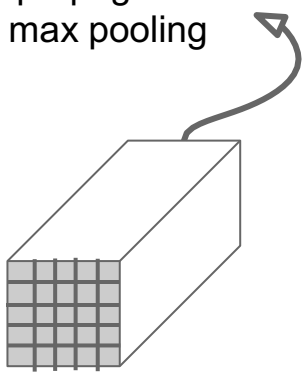
Hi-res input image:
3 x 800 x 600
with region
proposal

Convolution
and Pooling



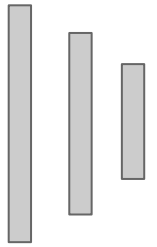
Hi-res conv features:
C x H x W
with region proposal

Can back propagate
similar to max pooling



RoI conv features:
C x h x w
for region proposal

Fully-connected
layers



Fully-connected layers expect
low-res conv features:
C x h x w

Fast R-CNN Results

Fast!

FASTER!

Better!

	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**

Using VGG-16 CNN on Pascal VOC 2007 dataset

Problem of Fast R-CNN?

- Test-time speeds don’t include region proposals

- SOLUTION: ?

Problem of Fast R-CNN?

- Computationally: Test-time speeds don't include region proposals

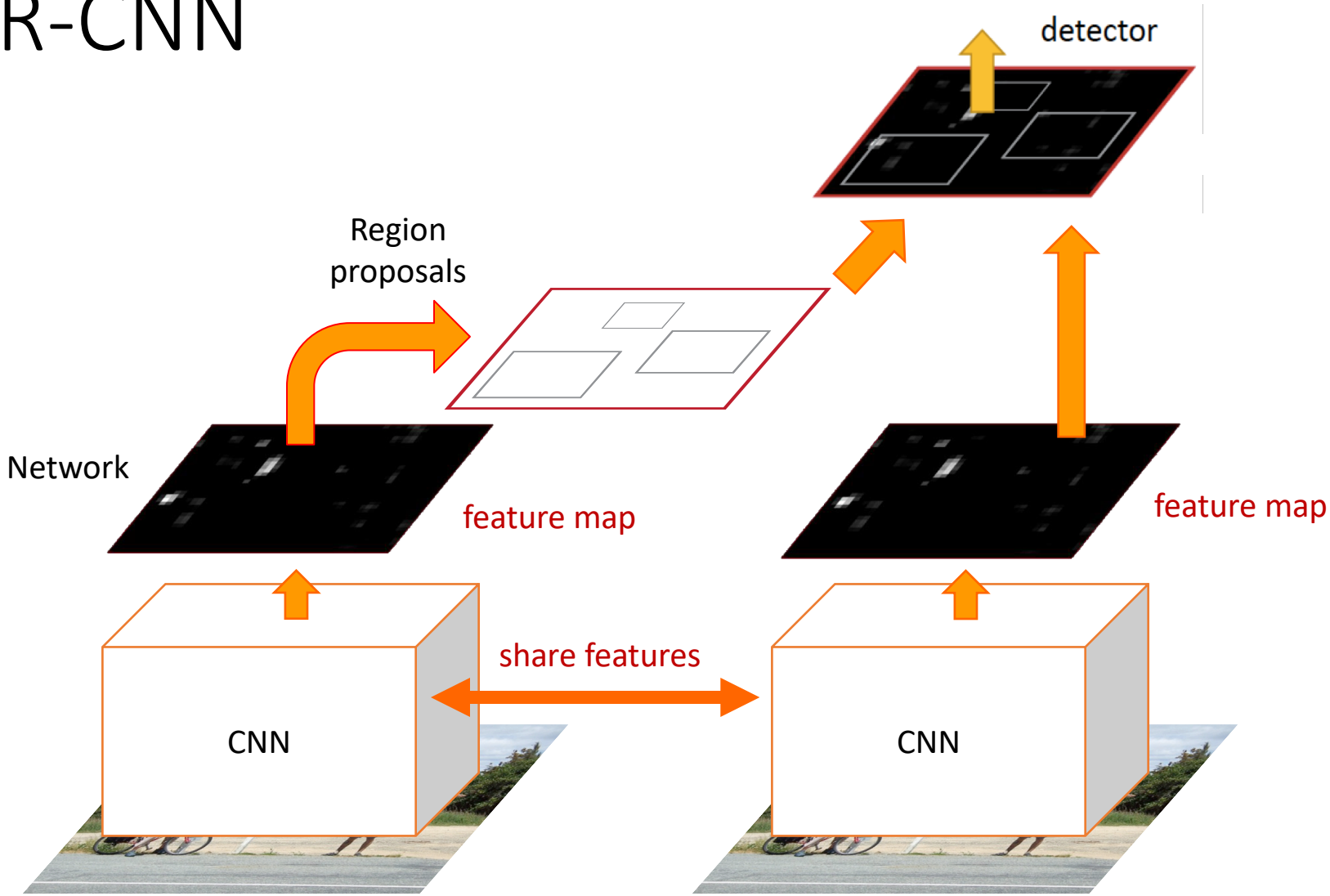
- SOLUTION: make the CNN do region proposals too!

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

Slide credit: Ross Girschick

Faster R-CNN

New: Region Proposal Network



S. Ren, K. He, R. Girshick, and J. Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS 2015

R-CNN Summary:

R-CNN:	Propose regions first. Classify proposed regions one at a time. Output contains: label + bounding box.
Fast R-CNN:	Propose regions after the convolutional net. Use convolution implementation of sliding windows to classify all the proposed regions. End-to-end.
Faster R-CNN:	Use ConvNet to propose regions. End-to-end.

[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]
[Girshik, 2015. Fast R-CNN]
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]

- Questions?

- Sliding Windows