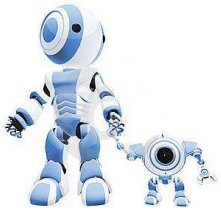
Convolutional Neural Networks for Object Recognition

Marius Leordeanu



LeMas-2016
Summer School

University Politehnica of Bucharest

Overview

- What is Computer Vision ?
- Convolutional Neural Networks
- **Convolutional Networks for Visual Object Recognition**

Based on the course materials and slides by Fei-Fei Li, Andrej Karpathy and Justin Johnson at Stanford University
http://cs231n.stanford.edu/syllabus.html

Learning to See

From Eyes …

to Vision

What is vision ?

What we see

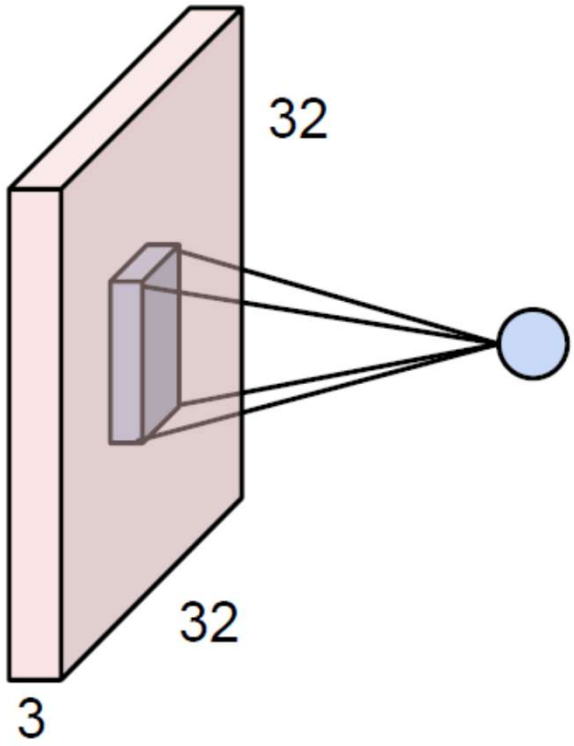
0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

What a computer sees

Vision is an inference problem
it is a way of thinking

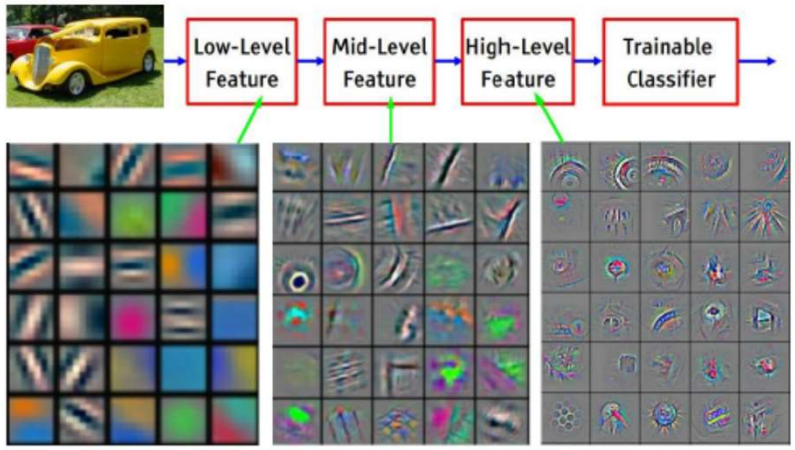
Many different 3D scenes could have given rise to the same 2D picture.

Convolution



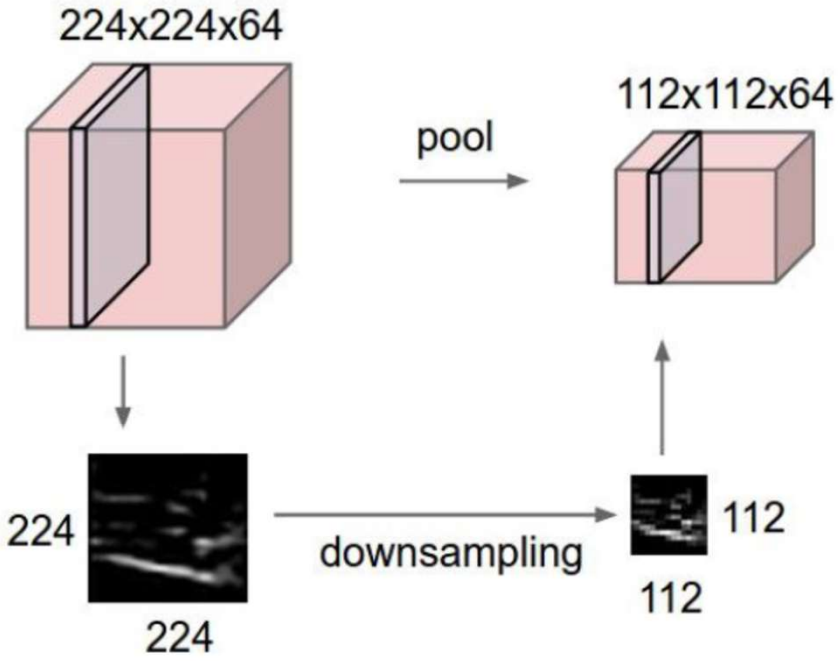
Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Pooling



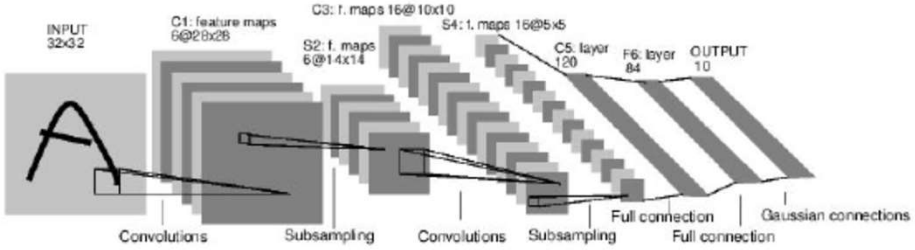
1	1	2	4
5	**6**	7	**8**
3	2	1	0
1	2	3	**4**

2x2 max pooling

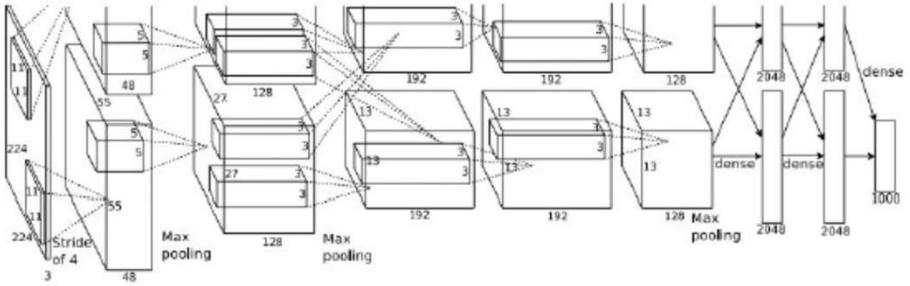
6	8
3	4

Case Studies

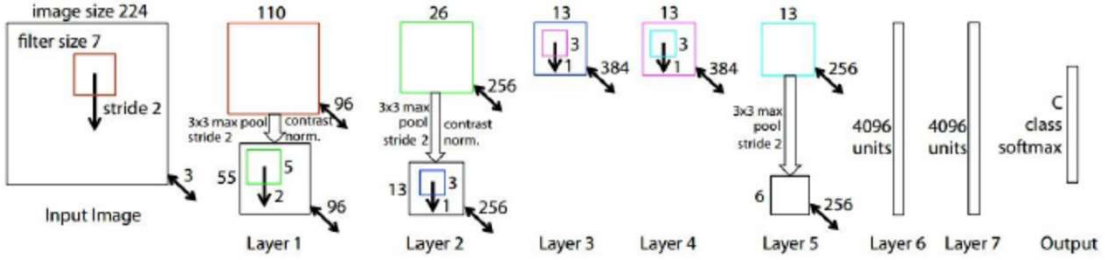
LeNet
(1998)



AlexNet
(2012)



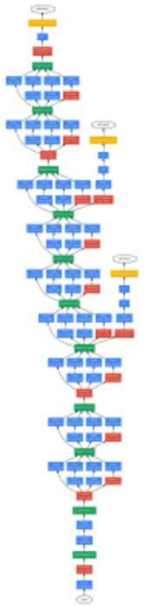
ZFNet
(2013)



Case Studies

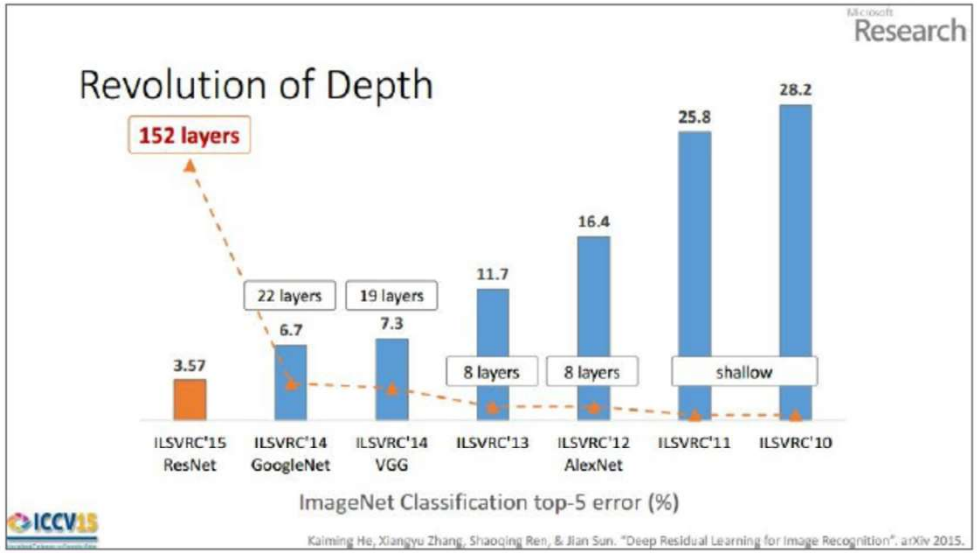
D	E
16 weight layers	19 weight layers
ge)	
conv3-64 conv3-64	conv3-64 conv3-64
conv3-128 conv3-128	conv3-128 conv3-128
conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool	
FC-4096	
FC-4096	
FC-1000	
soft-max	

VGG
(2014)

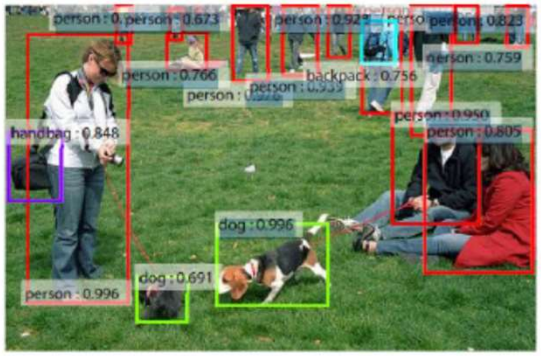


GoogLeNet
(2014)

ResNet
(2015)



Localization and Detection



Results from Faster R-CNN, Ren et al 2015

Computer Vision Tasks

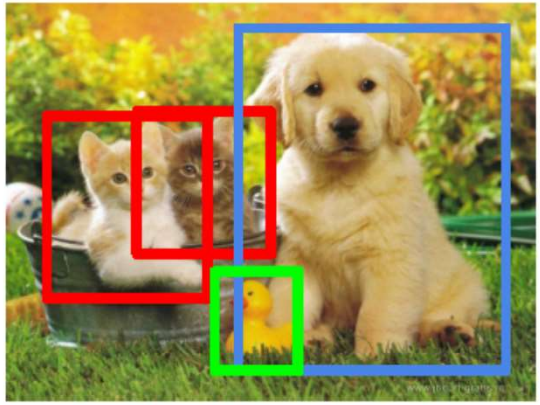
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation

Classification + Localization: Task

Classification: C classes
Input: Image
Output: Class label
Evaluation metric: Accuracy

CAT

Localization:
Input: Image
Output: Box in the image (x, y, w, h)
Evaluation metric: Intersection over Union

(x, y, w, h)

Classification + Localization: Do both

Idea #1: Localization as Regression

Input: image

Only one object,
simpler than detection

Neural Net

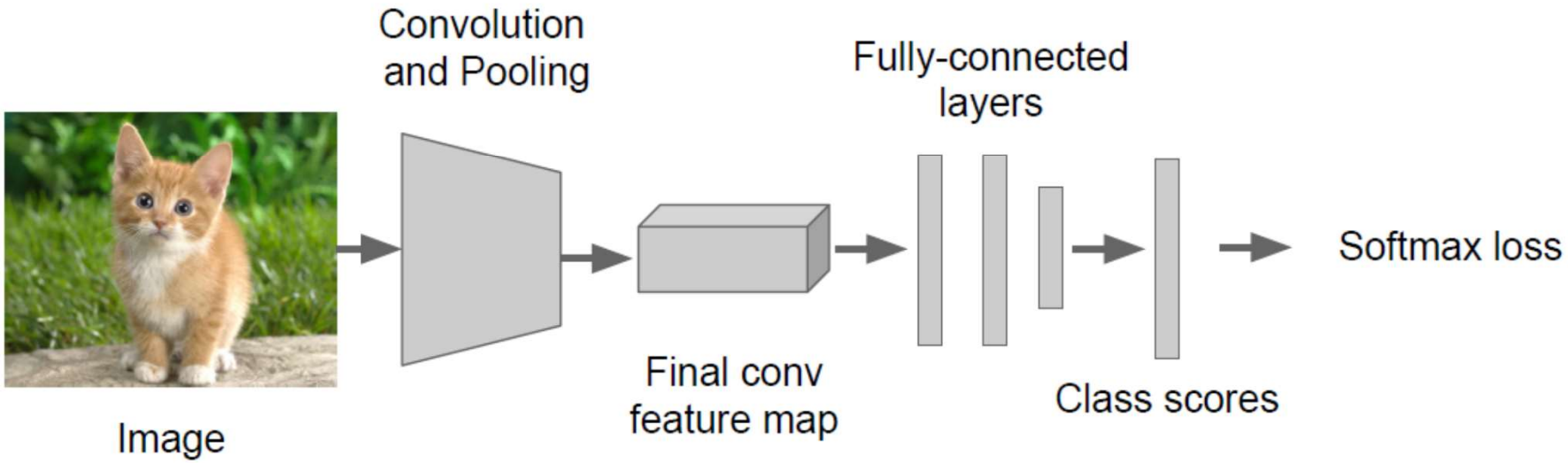
Output:
Box coordinates
(4 numbers)

Correct output:
box coordinates
(4 numbers)

Loss:
L2 distance

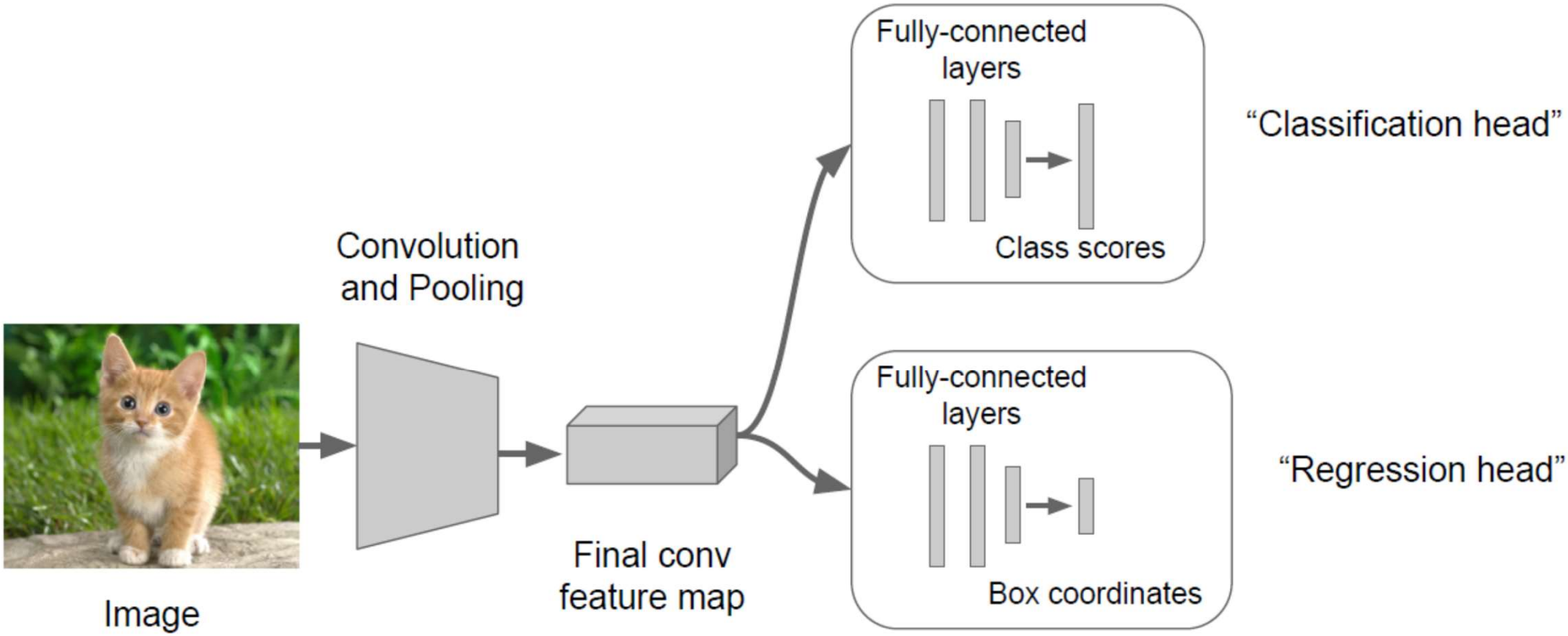
Simple Recipe for Classification + Localization

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



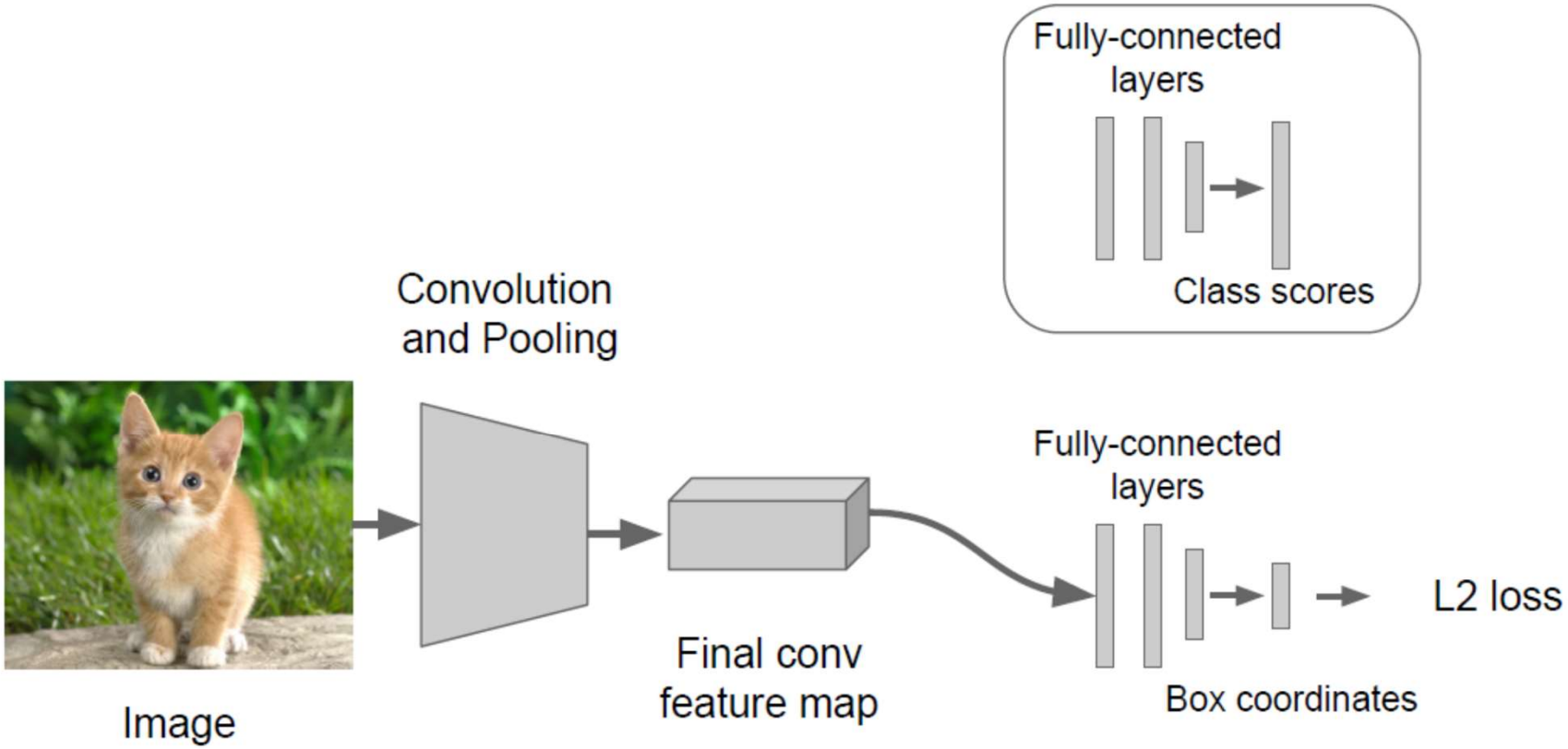
Simple Recipe for Classification + Localization

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



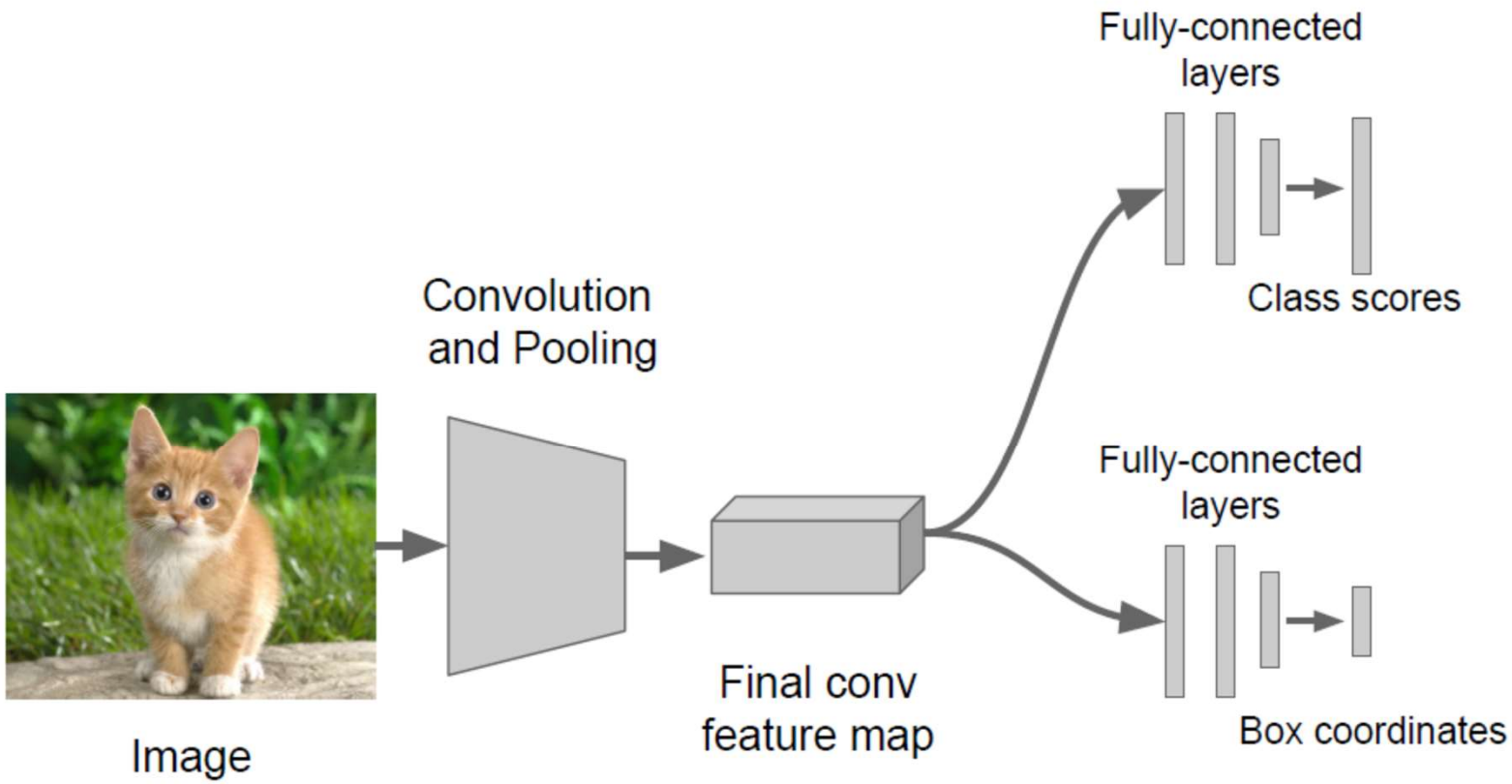
Simple Recipe for Classification + Localization

Step 3: Train the regression head only with SGD and L2 loss



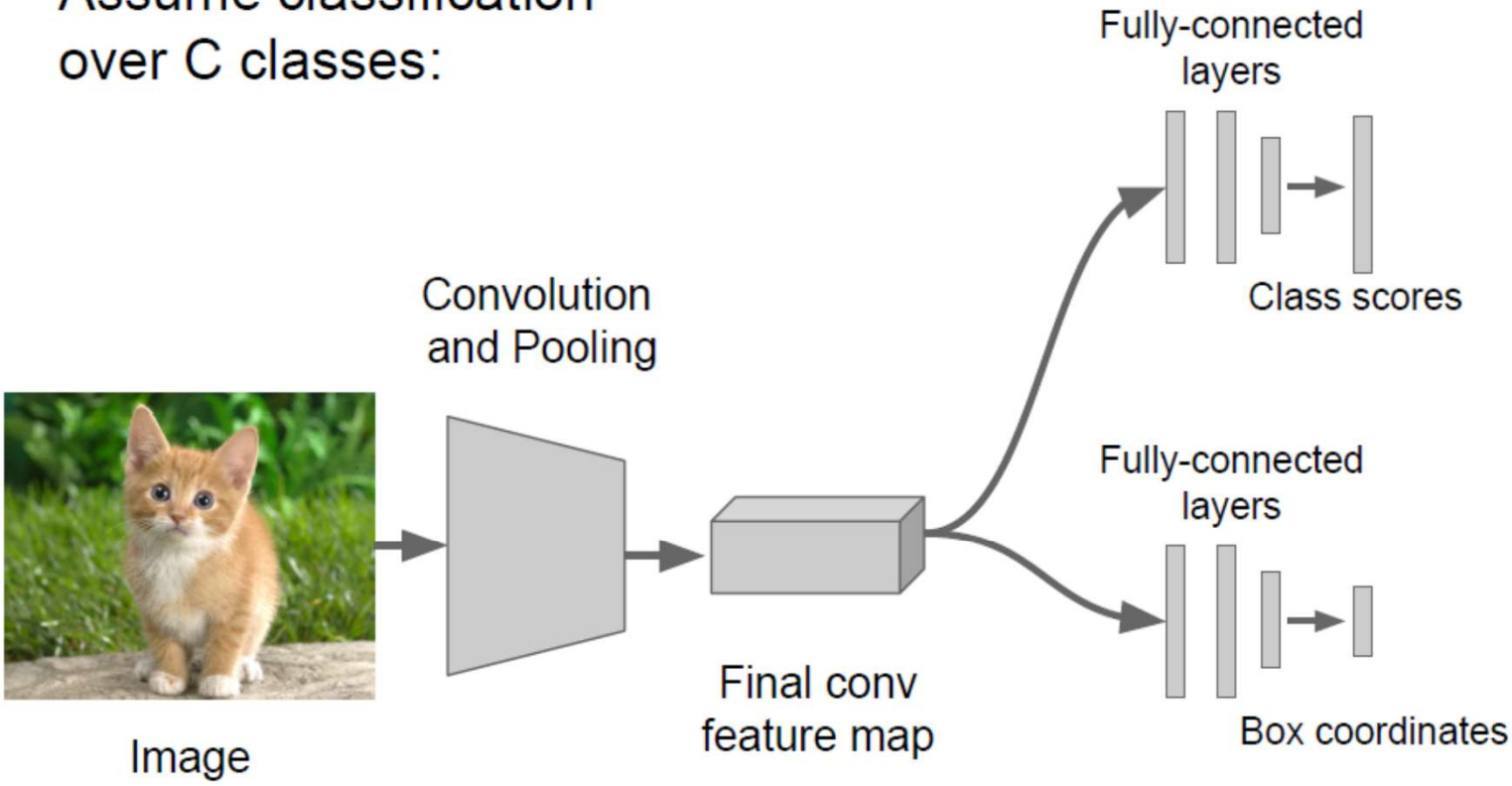
Simple Recipe for Classification + Localization

Step 4: At test time use both heads



Per-class vs class agnostic regression

Assume classification over C classes:

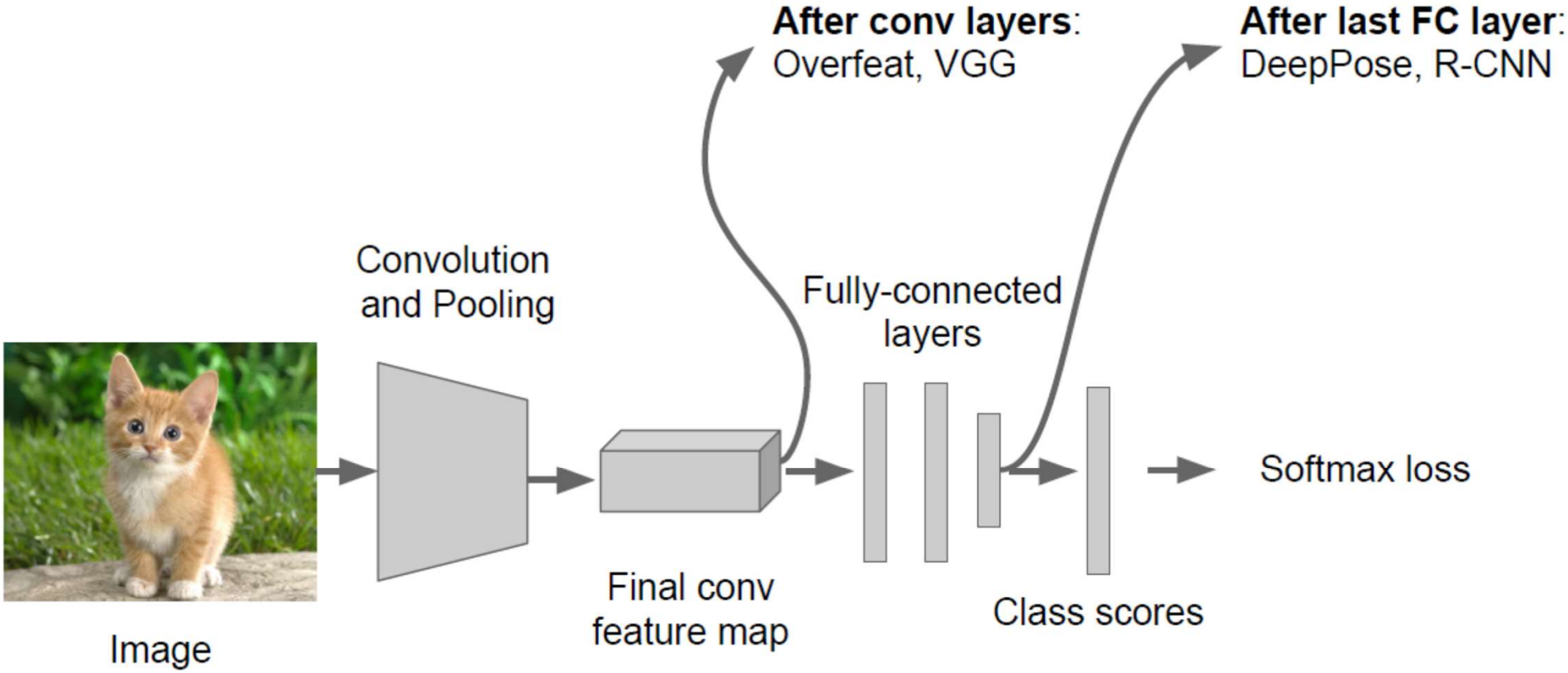


Classification head:
C numbers
(one per class)

Class agnostic:
4 numbers
(one box)

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

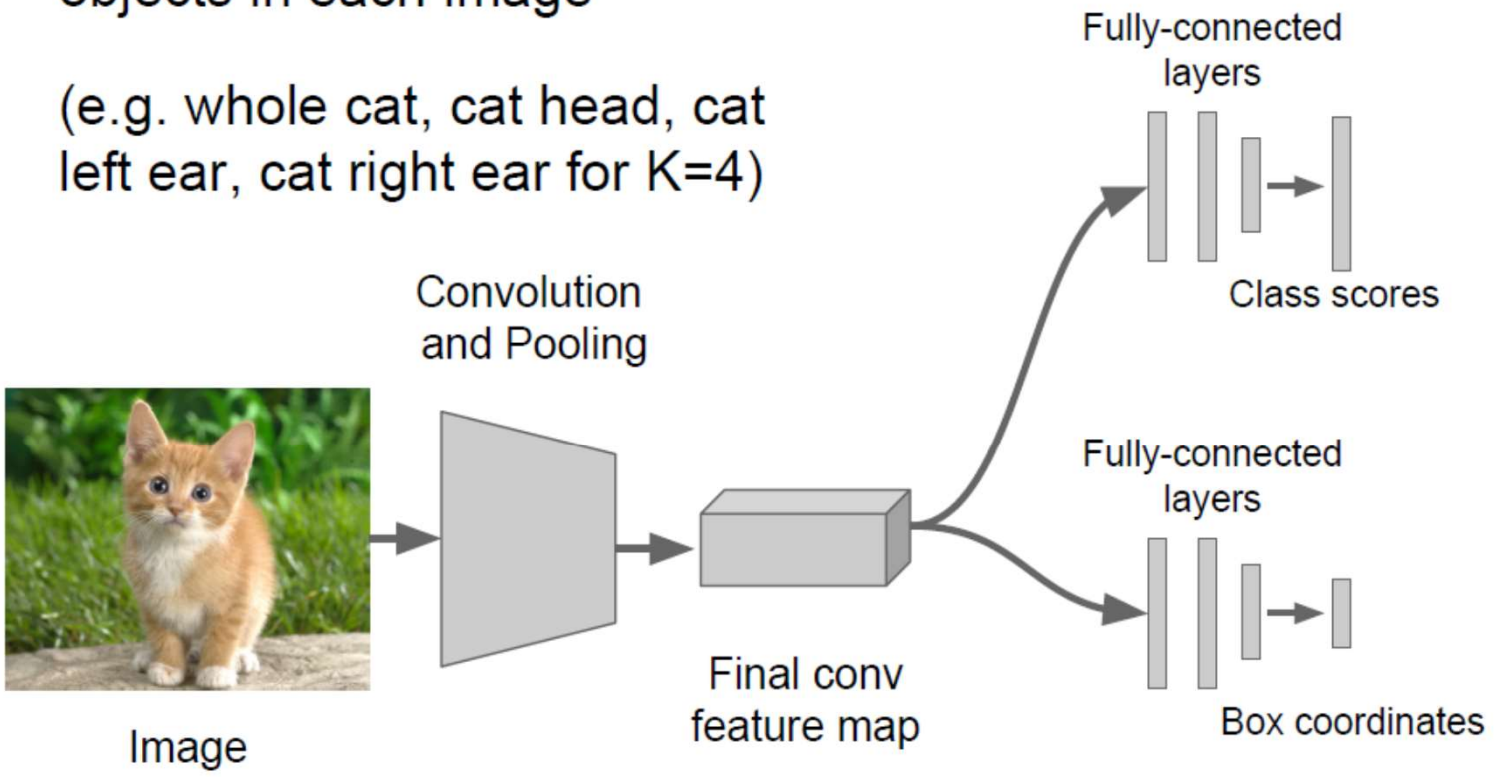
Where to attach the regression head?



Aside: Localizing multiple objects

Want to localize **exactly** K objects in each image

(e.g. whole cat, cat head, cat left ear, cat right ear for K=4)



K x 4 numbers
(one box per object)

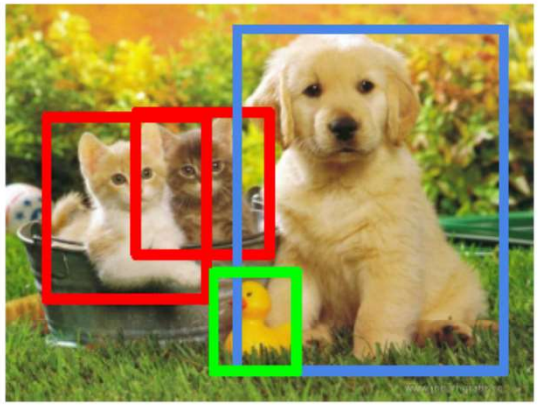
Computer Vision Tasks

Classification

Classification + Localization

Object Detection

Instance Segmentation



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

Problem: Need to test many positions and scales

Solution: If your classifier is fast enough, just do it

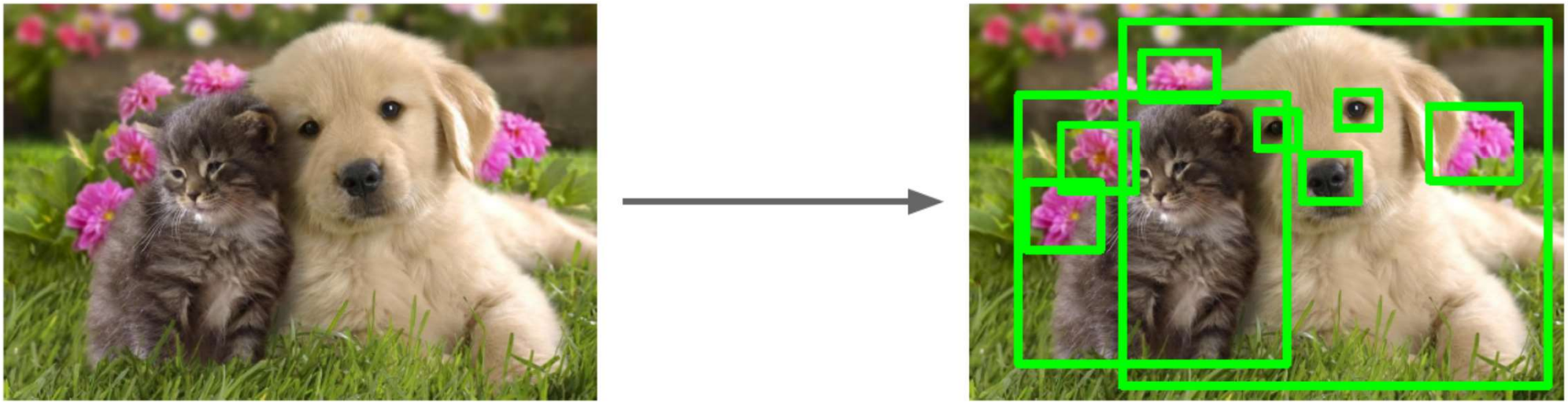
Detection as Classification

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

Solution: Only look at a tiny subset of possible positions

Region Proposals

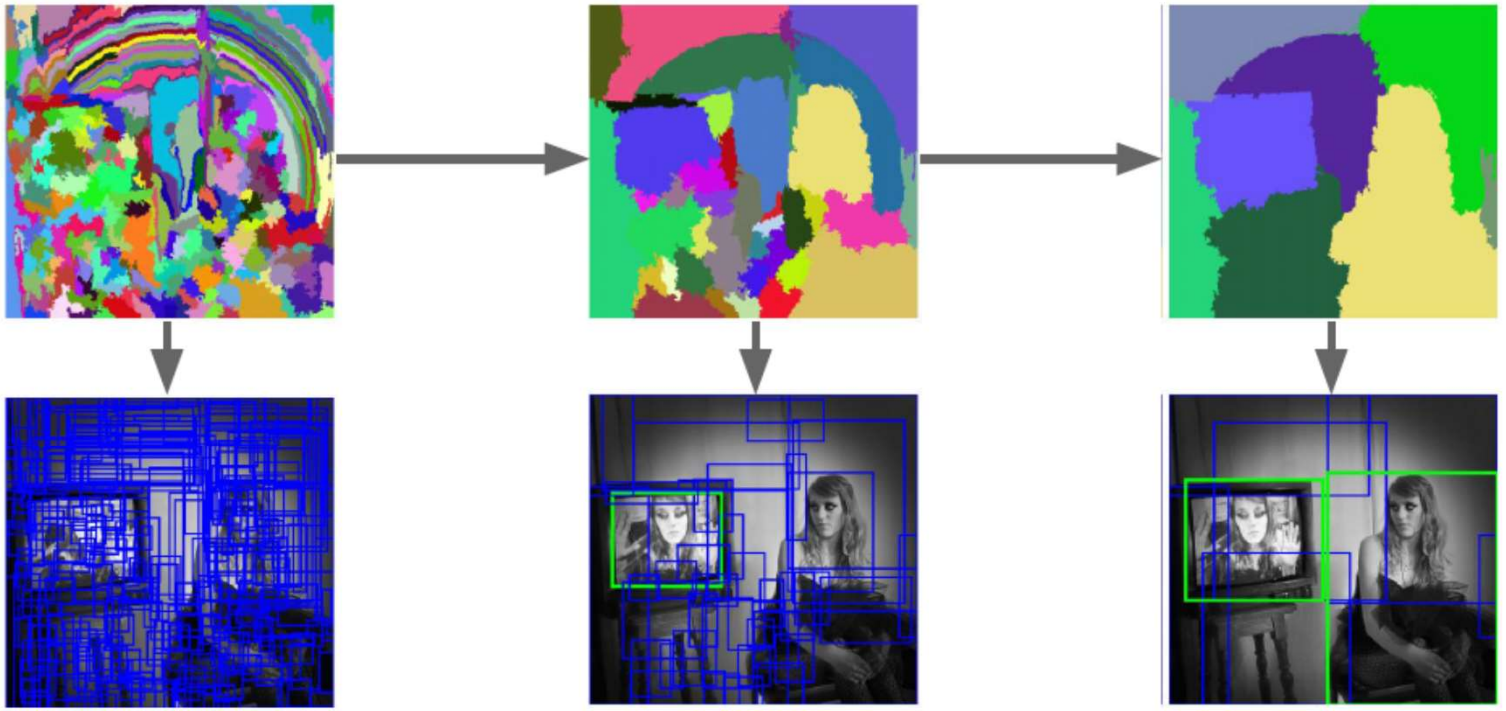
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes



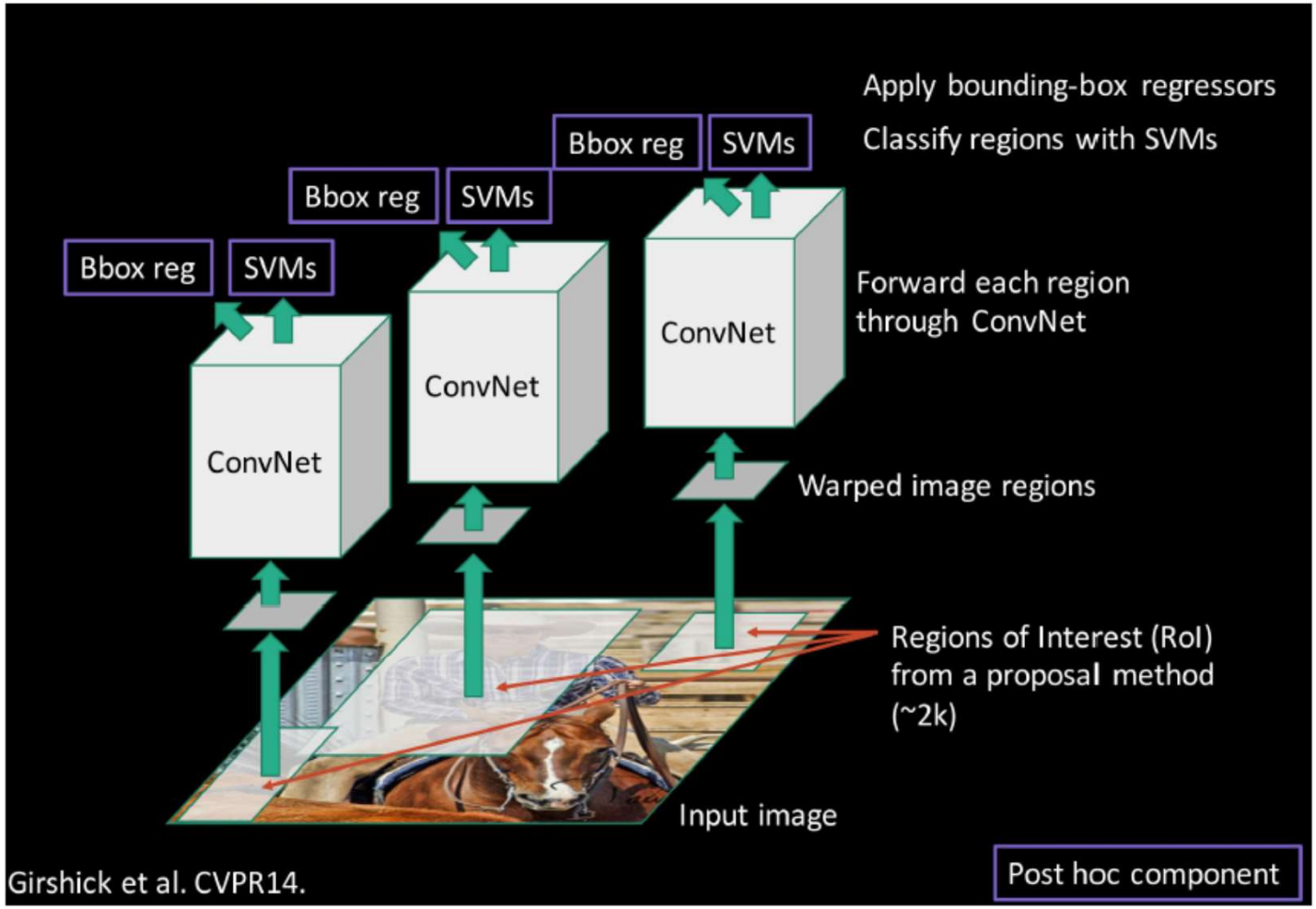
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

Region Proposals: Many other choices

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repea-tability	Recall Results	Detection Results
Bing [18]	Window scoring		✓	✓	0.2	⋆⋆⋆	⋆	·
CPMC [19]	Grouping	✓	✓	✓	250	-	⋆⋆	⋆
EdgeBoxes [20]	Window scoring		✓	✓	0.3	⋆⋆	⋆⋆⋆	⋆⋆⋆
Endres [21]	Grouping	✓	✓	✓	100	-	⋆⋆⋆	⋆⋆
Geodesic [22]	Grouping	✓		✓	1	⋆	⋆⋆⋆	⋆⋆
MCG [23]	Grouping	✓	✓	✓	30	⋆	⋆⋆⋆	⋆⋆⋆
Objectness [24]	Window scoring		✓	✓	3	·	⋆	·
Rahtu [25]	Window scoring		✓	✓	3	·	·	⋆
RandomizedPrim's [26]	Grouping	✓		✓	1	⋆	⋆	⋆⋆
Rantalankila [27]	Grouping	✓		✓	10	⋆⋆	·	⋆⋆
Rigor [28]	Grouping	✓		✓	10	⋆	⋆⋆	⋆⋆
SelectiveSearch [29]	Grouping	✓	✓	✓	10	⋆⋆	⋆⋆⋆	⋆⋆⋆
Gaussian				✓	0	·	·	⋆
SlidingWindow				✓	0	⋆⋆⋆	·	·
Superpixels		✓			1	⋆	·	·
Uniform				✓	0	·	·	·

Hosang et al, "What makes for effective detection proposals?", PAMI 2015

Putting it together: R-CNN

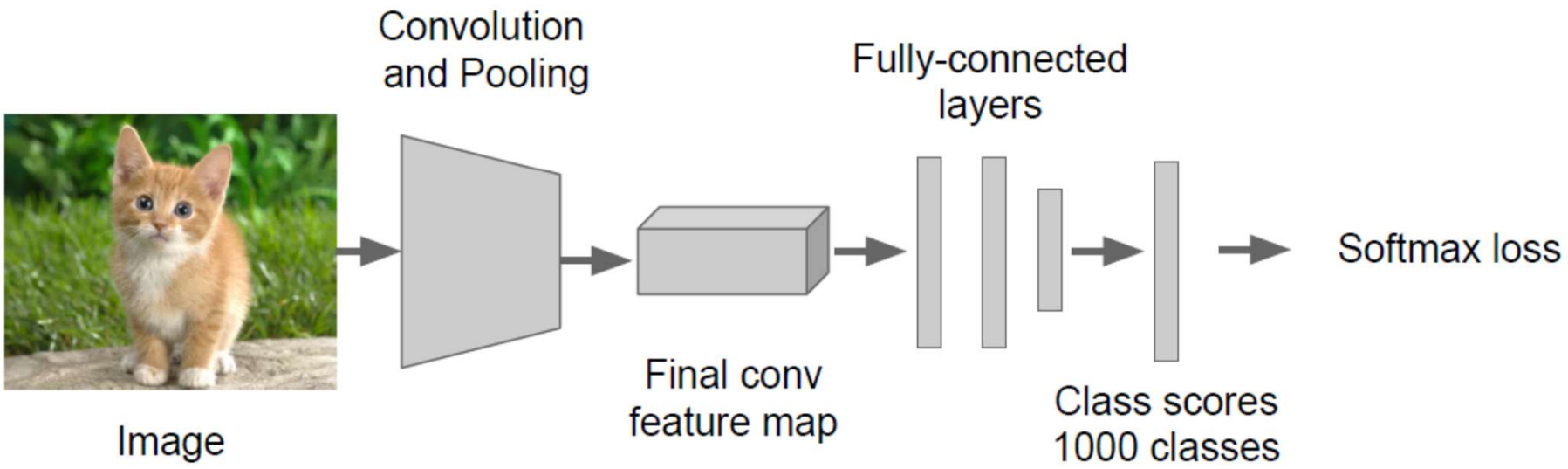


Girschick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

Slide credit: Ross Girschick

R-CNN Training

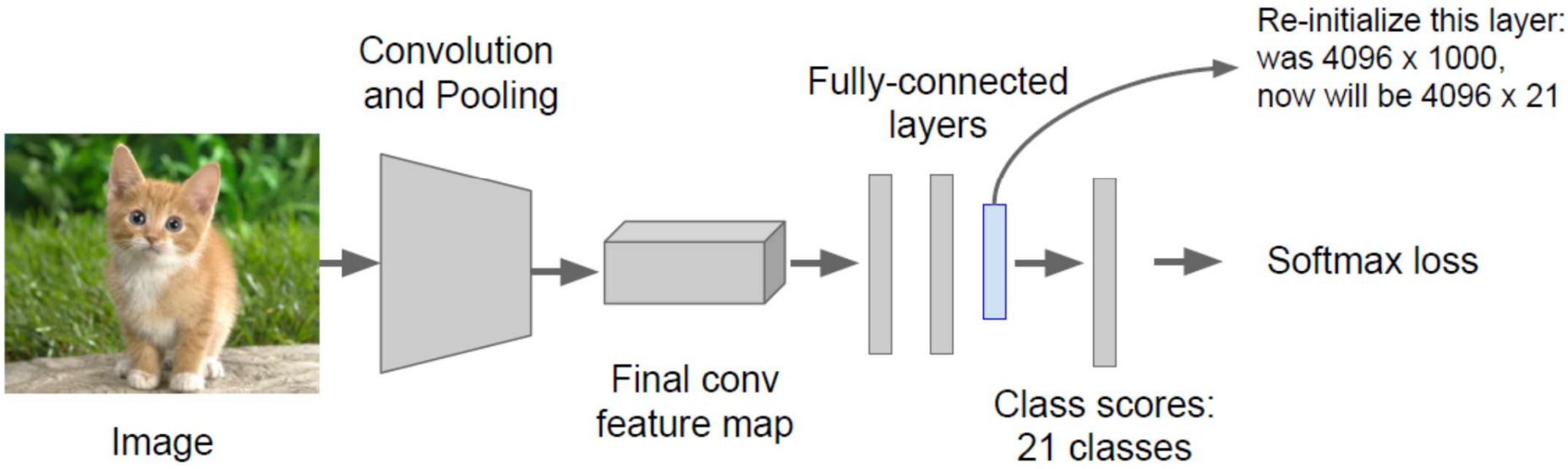
Step 1: Train (or download) a classification model for ImageNet (AlexNet)



R-CNN Training

Step 2: Fine-tune model for detection

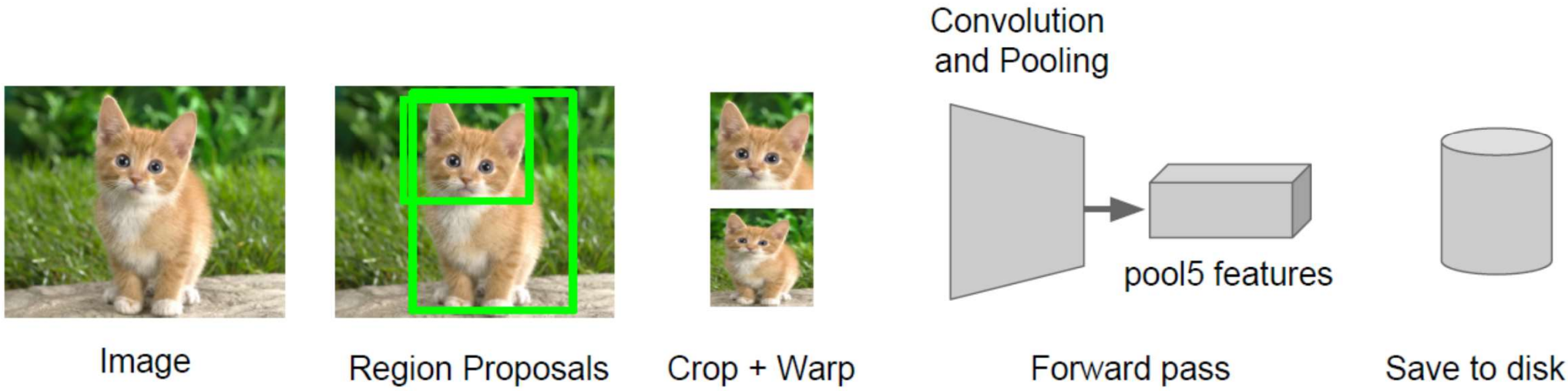
- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images



R-CNN Training

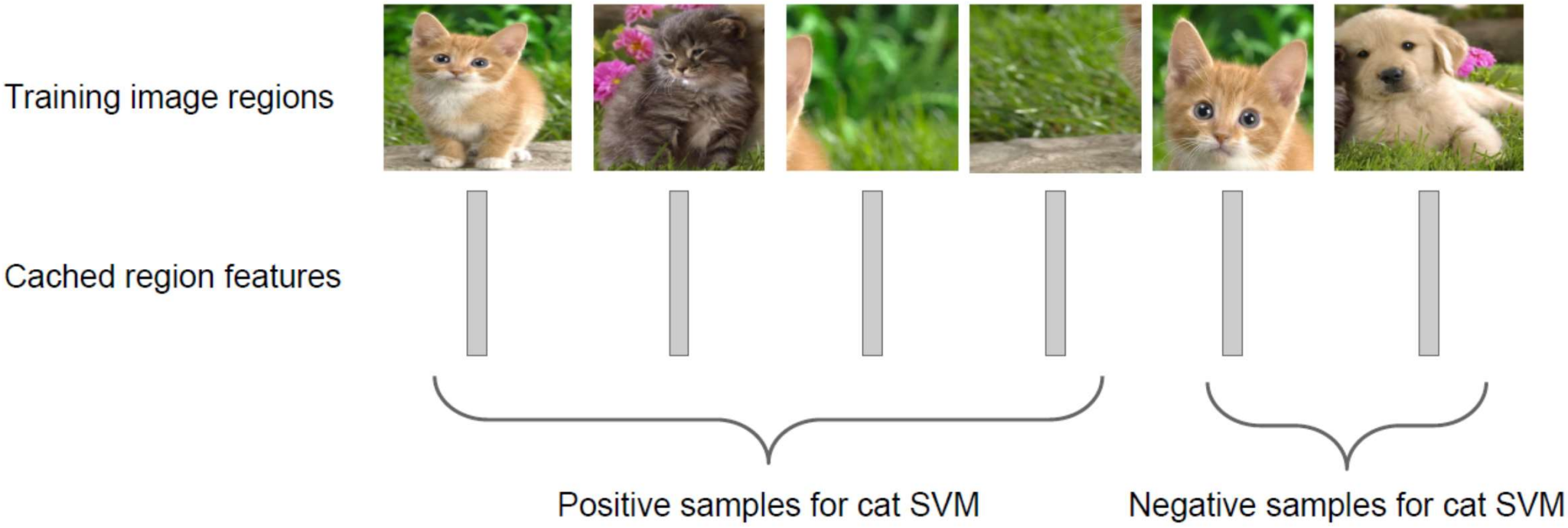
Step 3: Extract features

- 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!



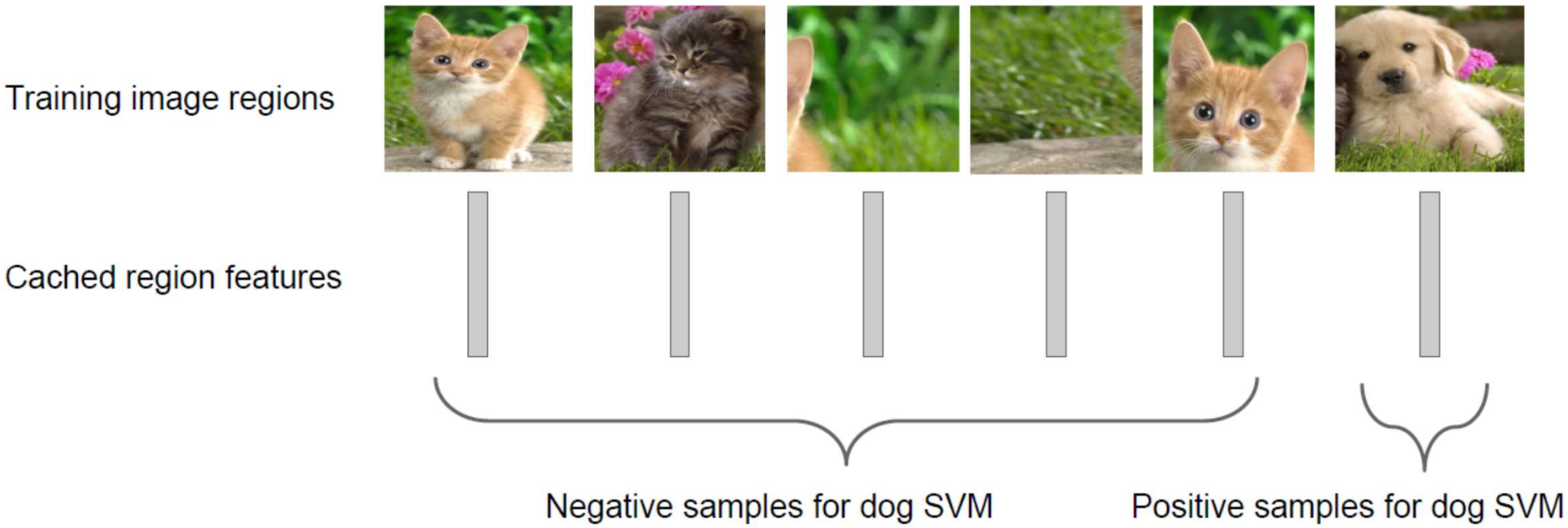
R-CNN Training

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



R-CNN Training

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



R-CNN Training

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

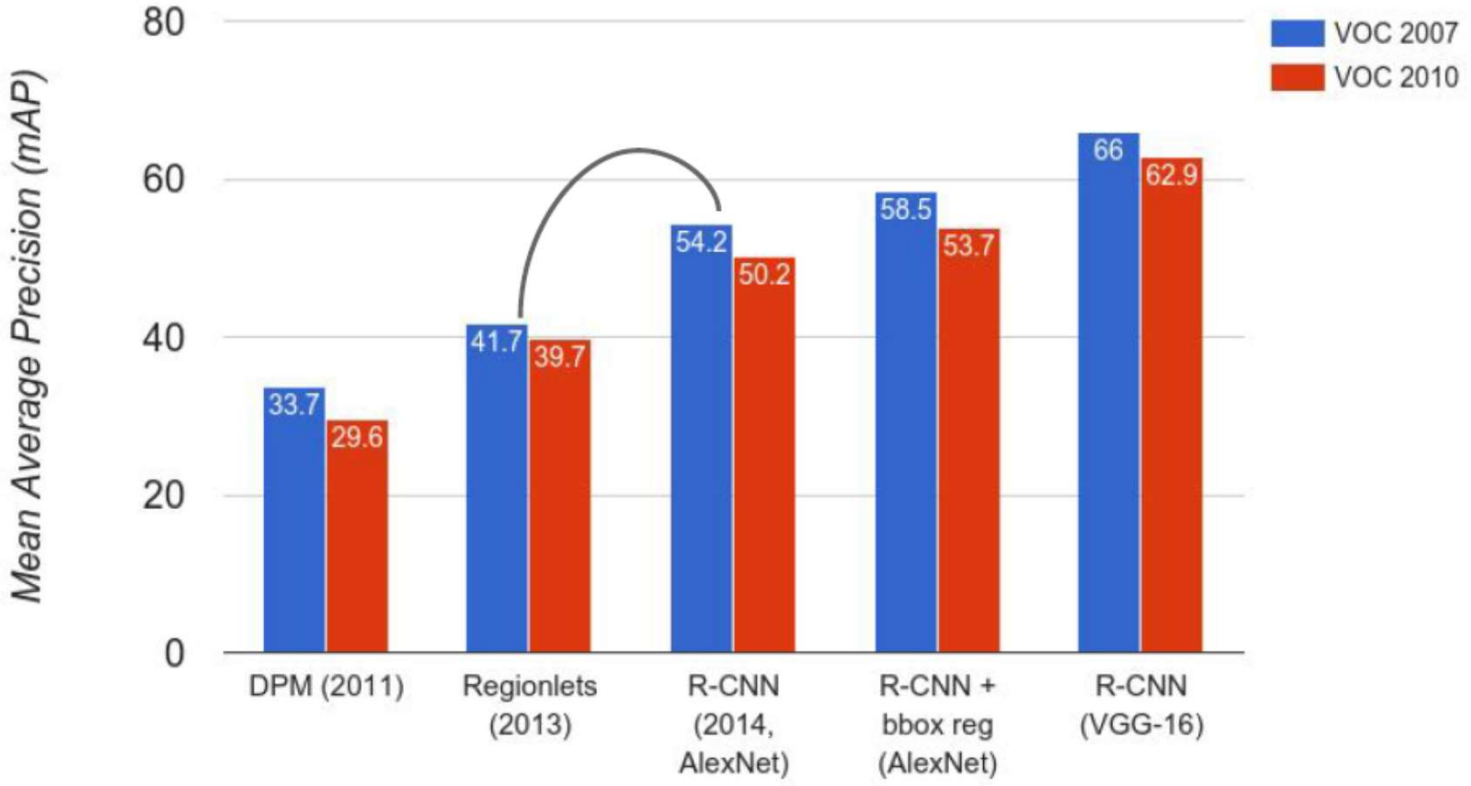
(0, 0, 0, 0)
Proposal is good

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

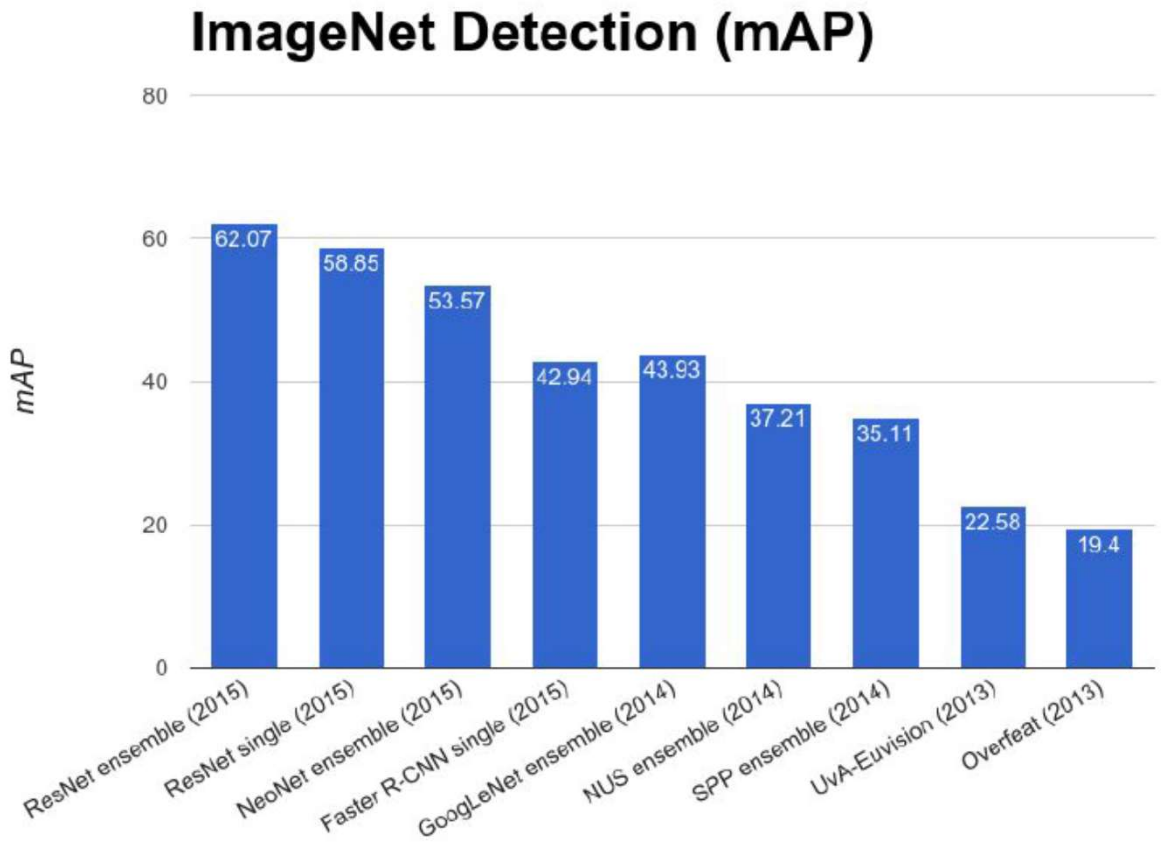
(0, 0, -0.125, 0)
Proposal too
wide

R-CNN Results

Big improvement compared to pre-CNN methods



ImageNet Detection 2013 - 2015



YOLO: You Only Look Once
Detection as Regression

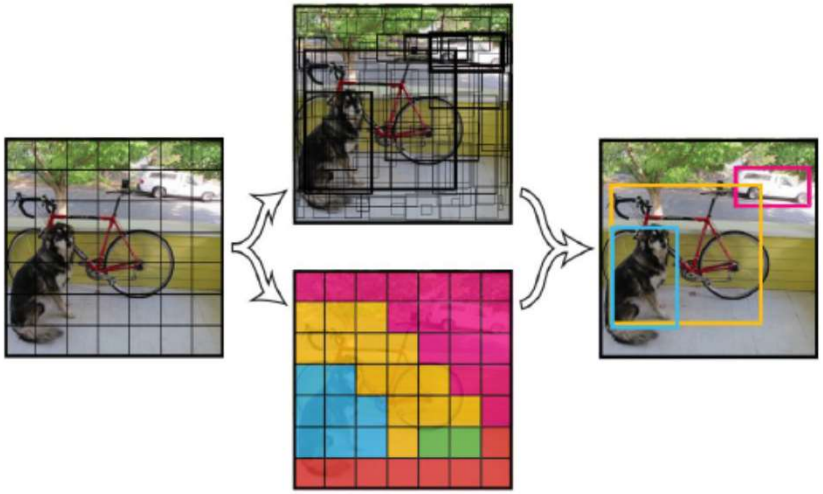
Divide image into S x S grid

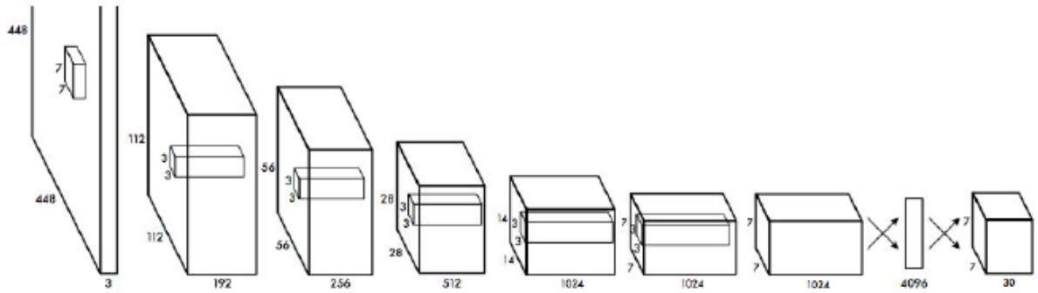
Within each grid cell predict:

- B Boxes: 4 coordinates + confidence
- Class scores: C numbers

Regression from image to
7 x 7 x (5 * B + C) tensor

Direct prediction using a CNN





Redmon et al, "You Only Look Once:
Unified, Real-Time Object Detection", arXiv 2015

YOLO: You Only Look Once
Detection as Regression

Faster than Faster R-CNN, but not as good

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [30]	2007	16.0	100
30Hz DPM [30]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [37]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[27]	2007+2012	73.2	7
Faster R-CNN ZF [27]	2007+2012	62.1	18

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", arXiv 2015

Object Detection code links:

R-CNN
(Cafffe + MATLAB): https://github.com/rbgirshick/rcnn
Probably don't use this; too slow

Fast R-CNN
(Caffe + MATLAB): https://github.com/rbgirshick/fast-rcnn

Faster R-CNN
(Caffe + MATLAB): https://github.com/ShaoqingRen/faster_rcnn
(Caffe + Python): https://github.com/rbgirshick/py-faster-rcnn

YOLO
http://pjreddie.com/darknet/yolo/
Maybe try this for projects?

Computational Frameworks for ConvNets

- Caffe
  http://caffe.berkeleyvision.org/
- Torch
  http://torch.ch/
- TensorFlow
  https://www.tensorflow.org/versions/r0.9/tutorials/deep_cnn/index.html
- Matconvnet
  http://www.vlfeat.org/matconvnet/

What is vision ?

We learn patterns from past visual experiences and recognize them now, to create our present visual world.