

# Machine Learning, Spring 2019

## Deep Learning for Object Localization and Detection

Instructor: Prof. Yi Fang

yfang@nyu.edu

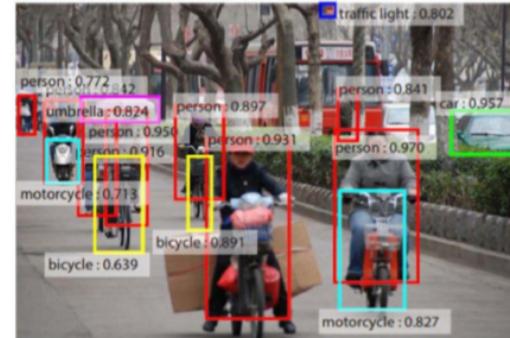
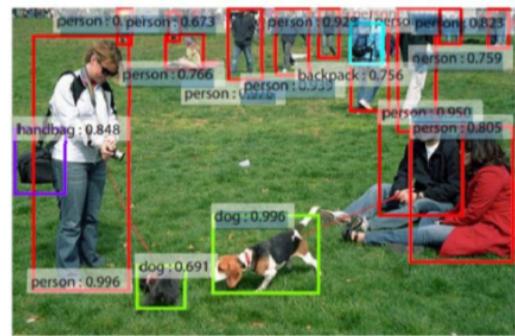
Reading Assignment: Chapter 10,11,12 & 13

Python tutorial: <http://learnpython.org/>

TensorFlow tutorial: <https://www.tensorflow.org/tutorials/>

PyTorch tutorial: <https://pytorch.org/tutorials/>

# Localization and Detection



Results from Faster R-CNN, Ren et al 2015

# Object Localization

## 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

# Classification + Localization: ImageNet

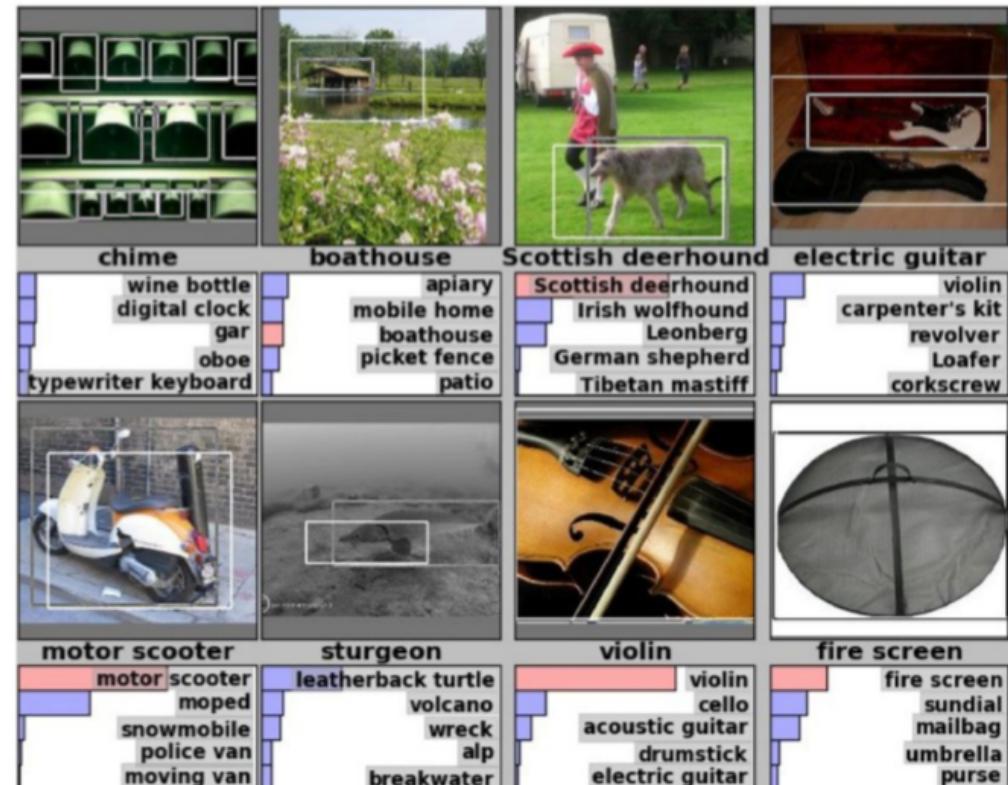
1000 classes (same as classification)

Each image has 1 class, at least one bounding box

~800 training images per class

Algorithm produces 5 (class, box) guesses

Example is correct if at least one guess has correct class AND bounding box at least 0.5 intersection over union (IoU)



Krizhevsky et. al. 2012

# Localization as Regression

**Input:** image



Neural Net  
→

**Output:**

Box coordinates  
(4 numbers)



**Loss:**

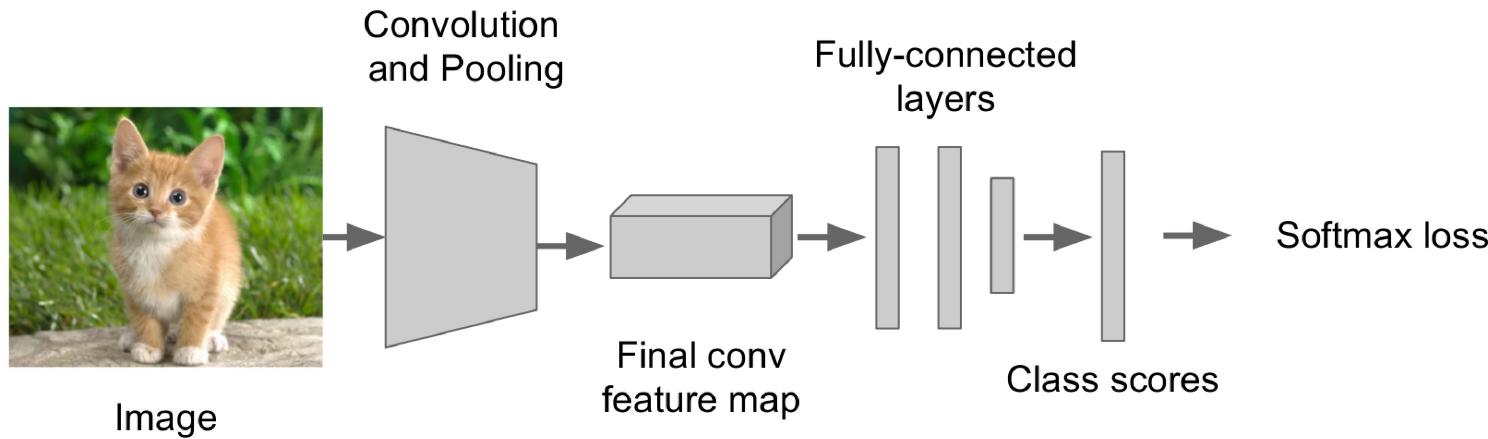
L2 distance

**Correct output:**  
box coordinates  
(4 numbers)

Only one object,  
simpler than detection

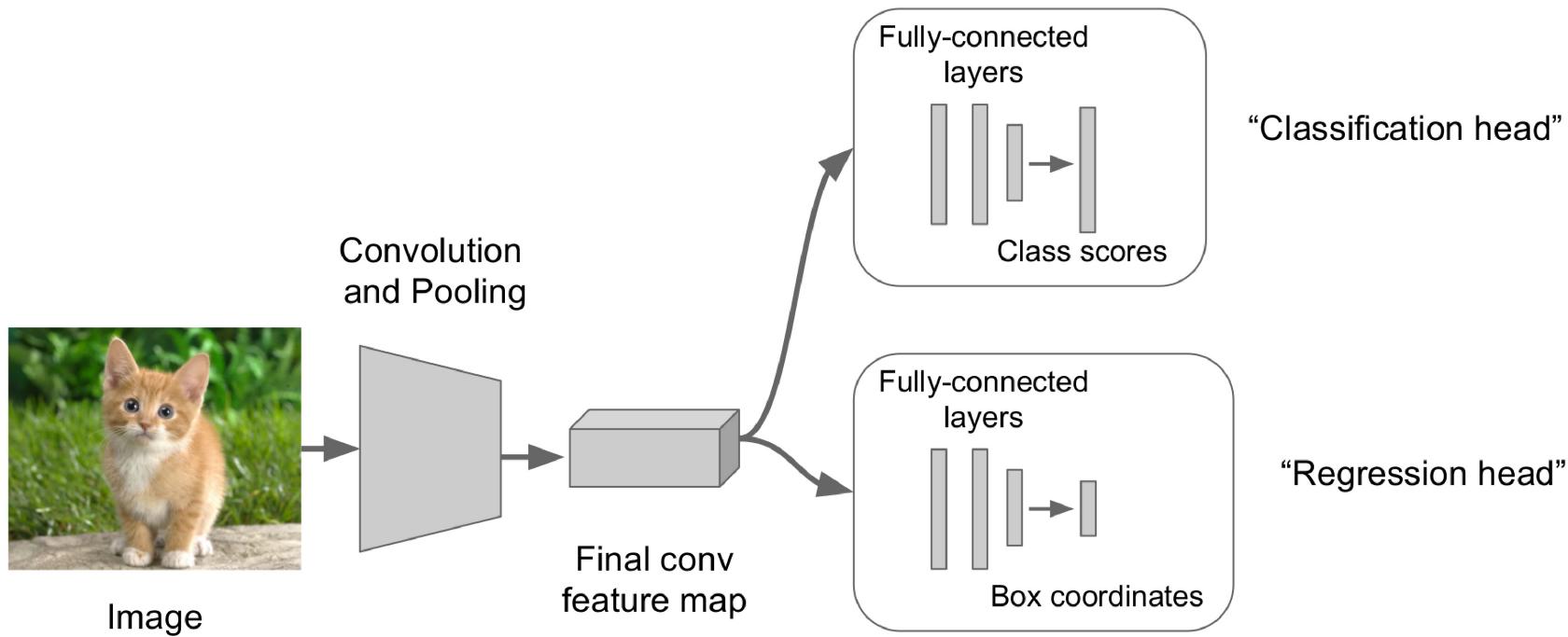
# Simple Recipe for Classification + Localization

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



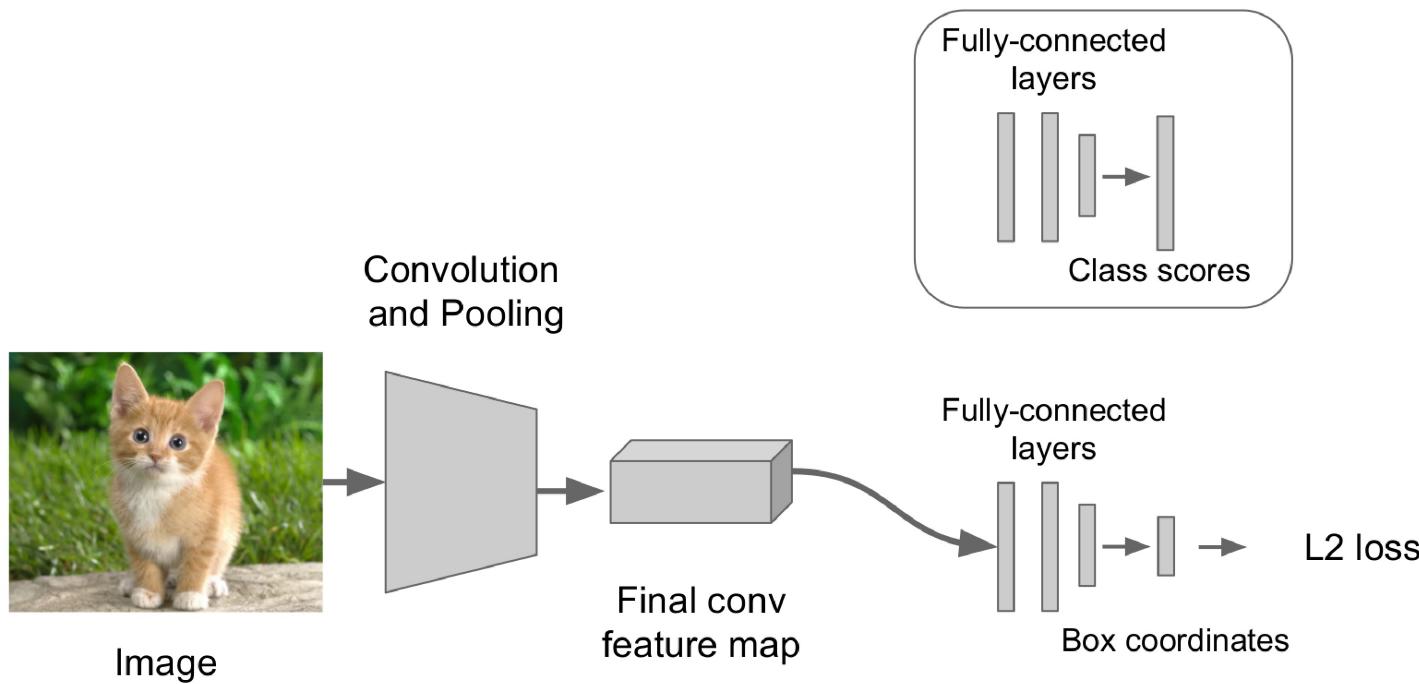
# Simple Recipe for Classification + Localization

**Step 2:** Attach new fully-connected “regression head” to the network



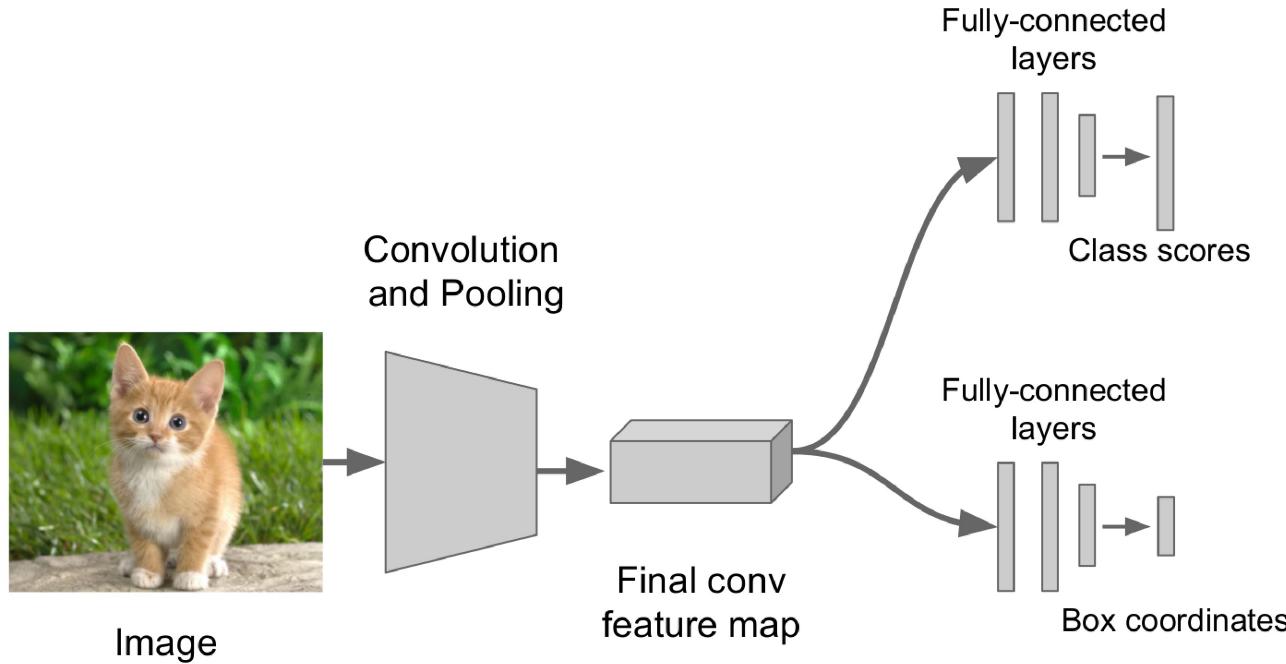
# Simple Recipe for Classification + Localization

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



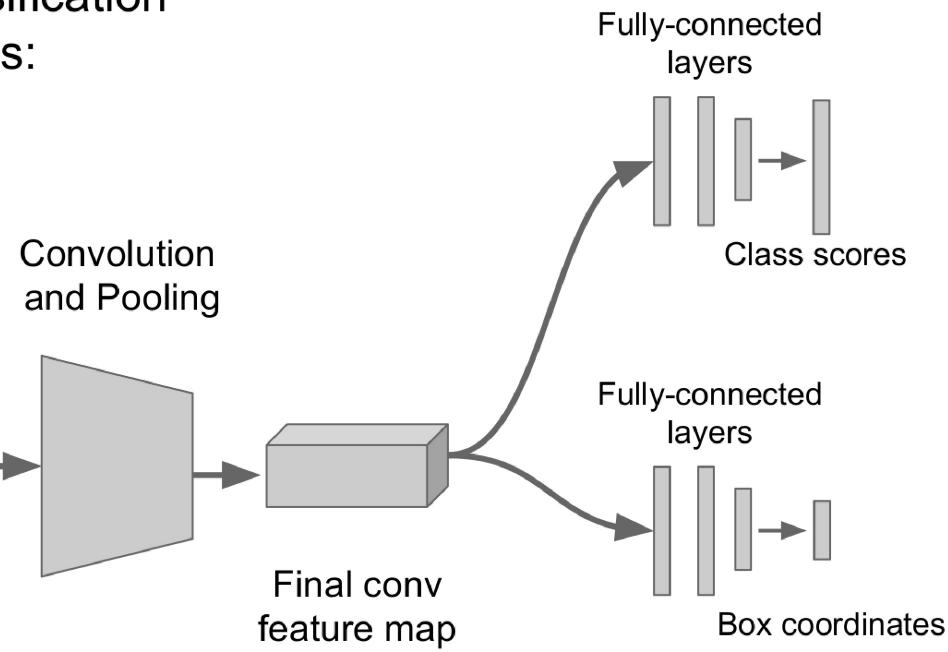
# Simple Recipe for Classification + Localization

**Step 4:** At test time use both heads



# Per-class vs class agnostic regression

Assume classification over C classes:

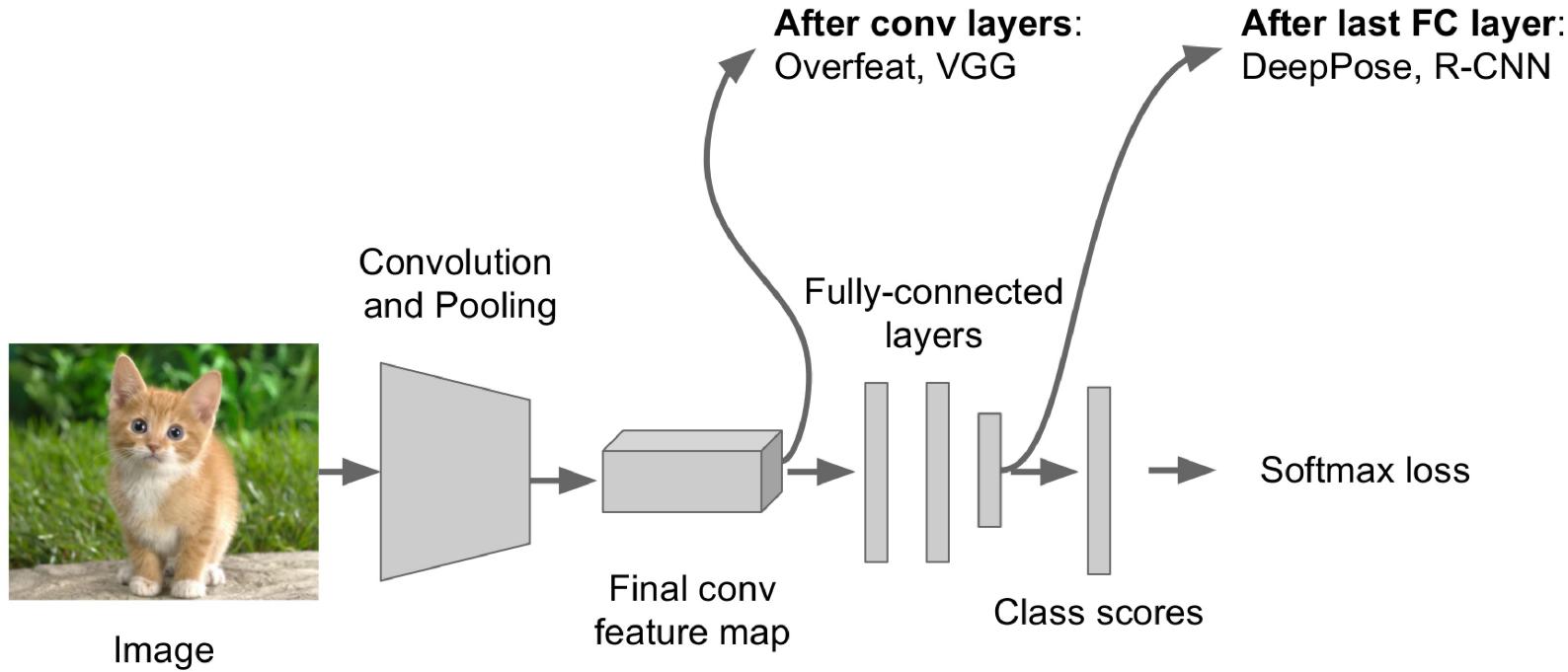


**Classification head:**  
C numbers  
(one per class)

**Class agnostic:**  
4 numbers  
(one box)

**Class specific:**  
 $C \times 4$  numbers  
(one box per class)

# Where to attach the regression head?

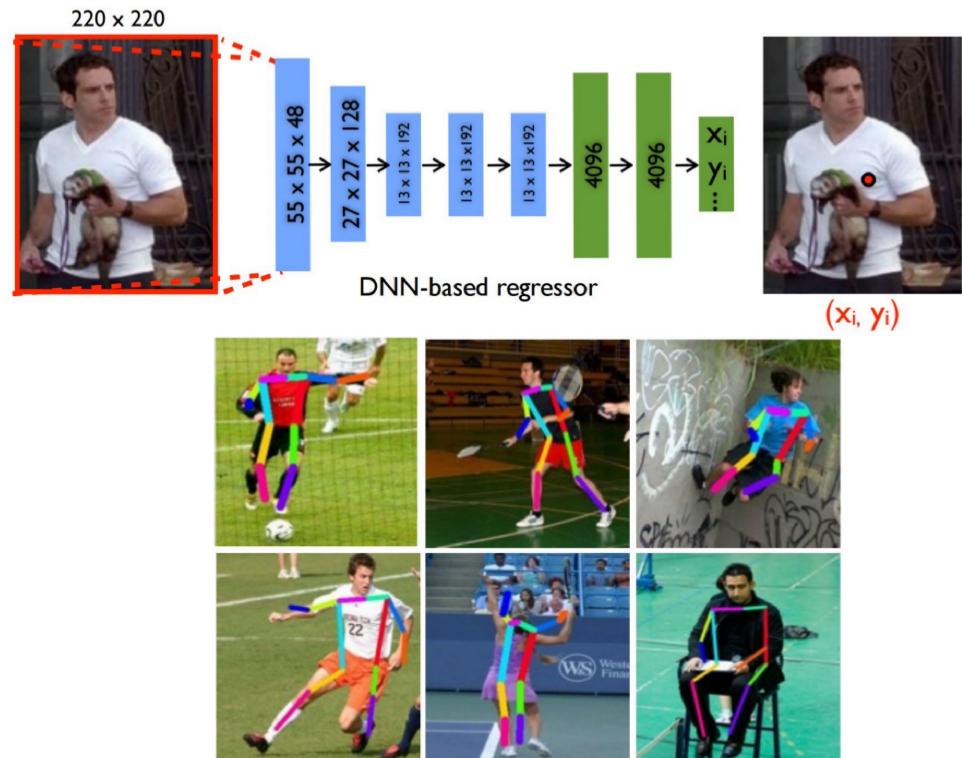


## Aside: Human Pose Estimation

Represent a person by K joints

Regress ( $x$ ,  $y$ ) for each joint from last fully-connected layer of AlexNet

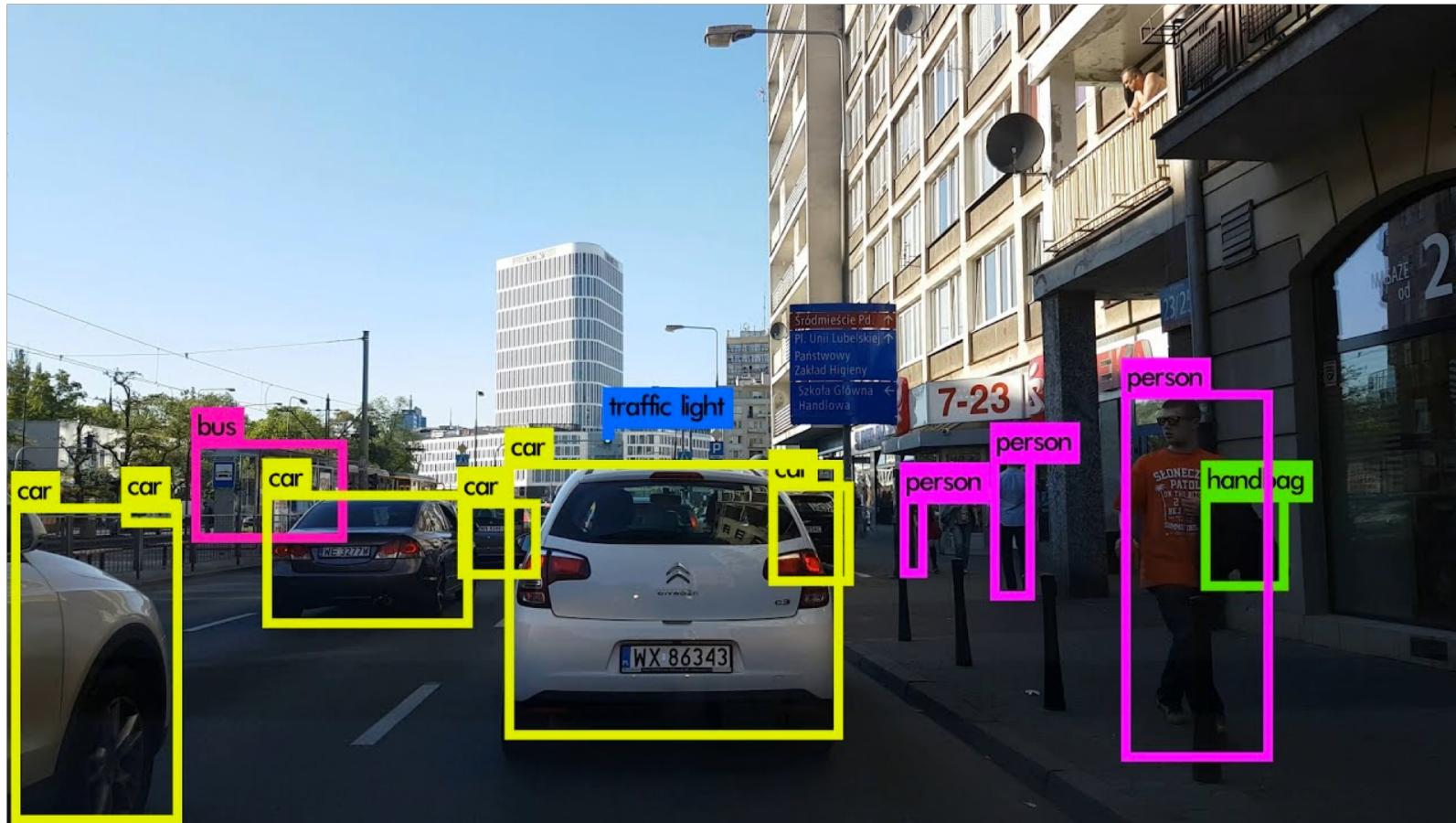
(Details: Normalized coordinates, iterative refinement)



Toshev and Szegedy, "DeepPose: Human Pose Estimation via Deep Neural Networks", CVPR 2014

- Localization as Regression
- Simple and Straightforward
- What is the issue?

# Multiple Objects in one Image



Src: <https://blog.paperspace.com/tag/series-yolo/>

Prof. Yi Fang (yfang@nyu.edu)

# Solution

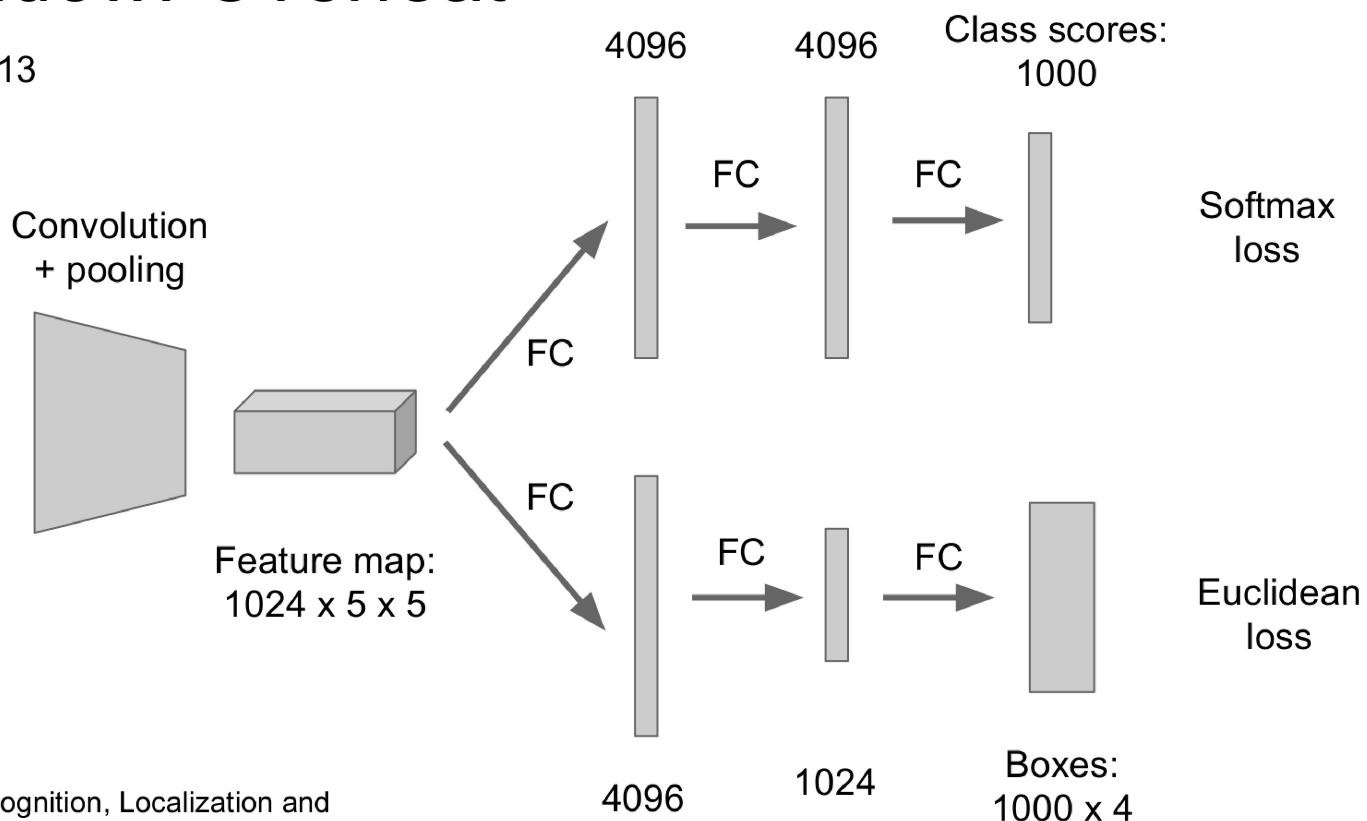
- Sliding Windows at multiple locations with multiple window size

# Sliding Window: Overfeat

Winner of ILSVRC 2013  
localization challenge



Image:  
 $3 \times 221 \times 221$



Sermanet et al, "Integrated Recognition, Localization and Detection using Convolutional Networks", ICLR 2014

# Sliding Window: Overfeat

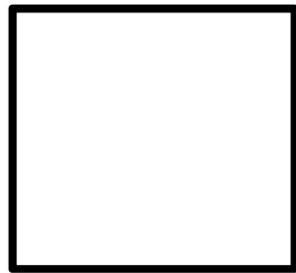


Network input:  
 $3 \times 221 \times 221$

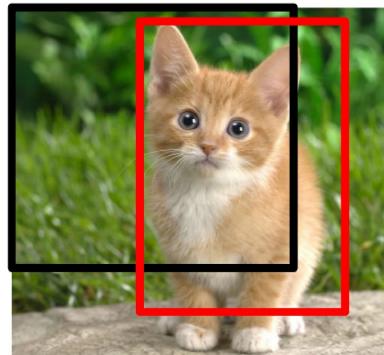


Larger image:  
 $3 \times 257 \times 257$

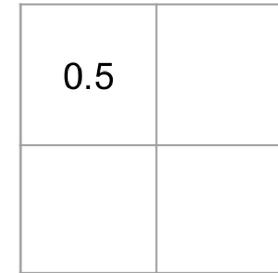
# Sliding Window: Overfeat



Network input:  
 $3 \times 221 \times 221$

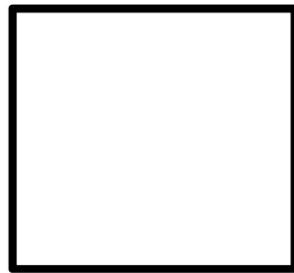


Larger image:  
 $3 \times 257 \times 257$

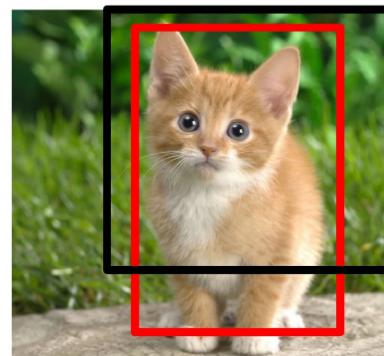


Classification scores:  
 $P(\text{cat})$

# Sliding Window: Overfeat



Network input:  
 $3 \times 221 \times 221$

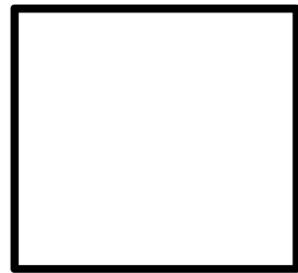


Larger image:  
 $3 \times 257 \times 257$

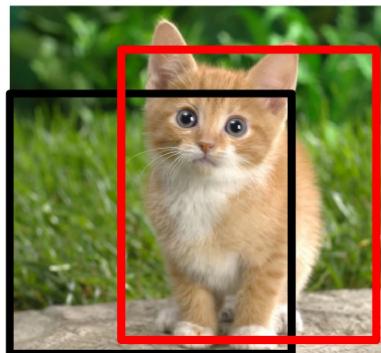
0.5	0.75

Classification scores:  
 $P(\text{cat})$

# Sliding Window: Overfeat



Network input:  
 $3 \times 221 \times 221$

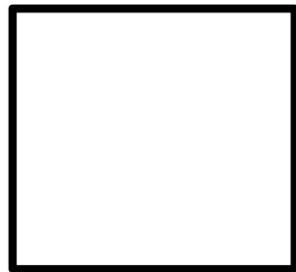


Larger image:  
 $3 \times 257 \times 257$

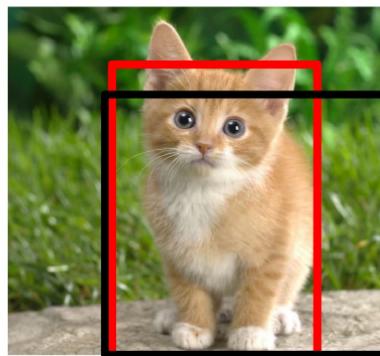
0.5	0.75
0.6	

Classification scores:  
 $P(\text{cat})$

# Sliding Window: Overfeat



Network input:  
 $3 \times 221 \times 221$

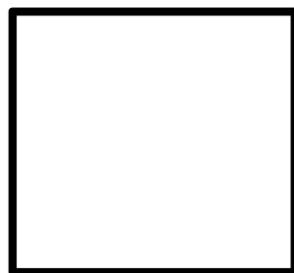


Larger image:  
 $3 \times 257 \times 257$

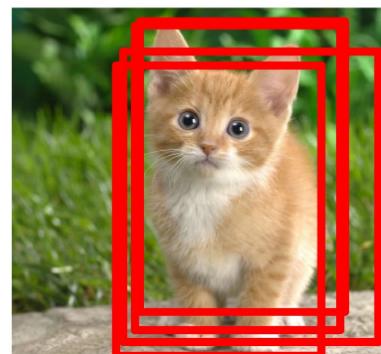
0.5	0.75
0.6	0.8

Classification scores:  
 $P(\text{cat})$

# Sliding Window: Overfeat



Network input:  
3 x 221 x 221

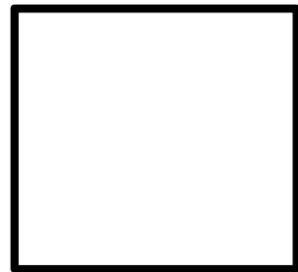


Larger image:  
3 x 257 x 257

0.5	0.75
0.6	0.8

Classification scores:  
 $P(\text{cat})$

# Sliding Window: Overfeat



Network input:  
 $3 \times 221 \times 221$



Larger image:  
 $3 \times 257 \times 257$

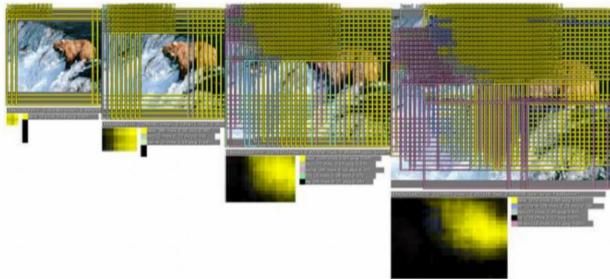
0.8

Classification score: P  
(cat)

# Sliding Window: Overfeat

In practice use many sliding window locations and multiple scales

Window positions + score maps



Box regression outputs

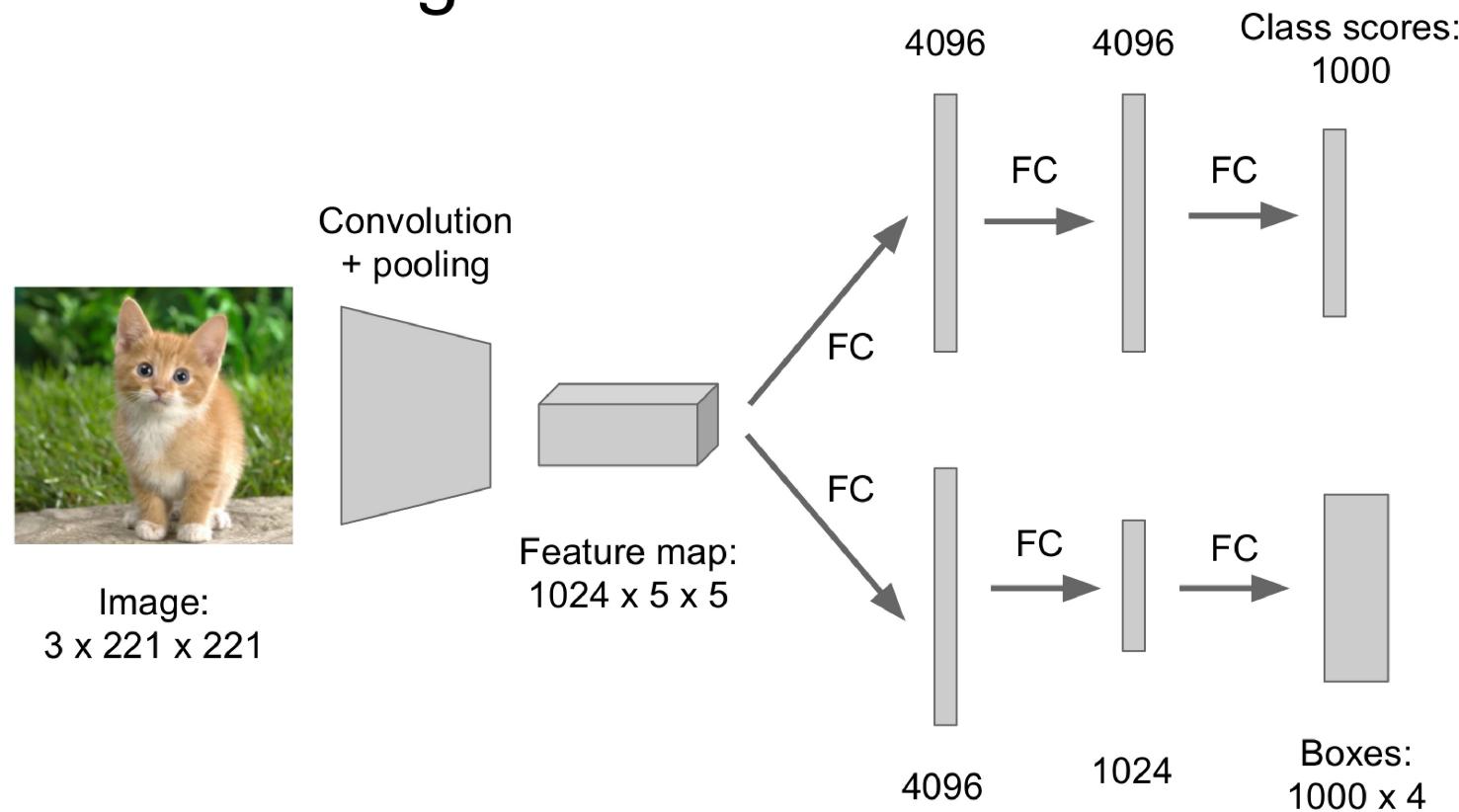


Final Predictions



Sermanet et al, "Integrated Recognition, Localization and Detection using Convolutional Networks", ICLR 2014

# Efficient Sliding Window: Overfeat

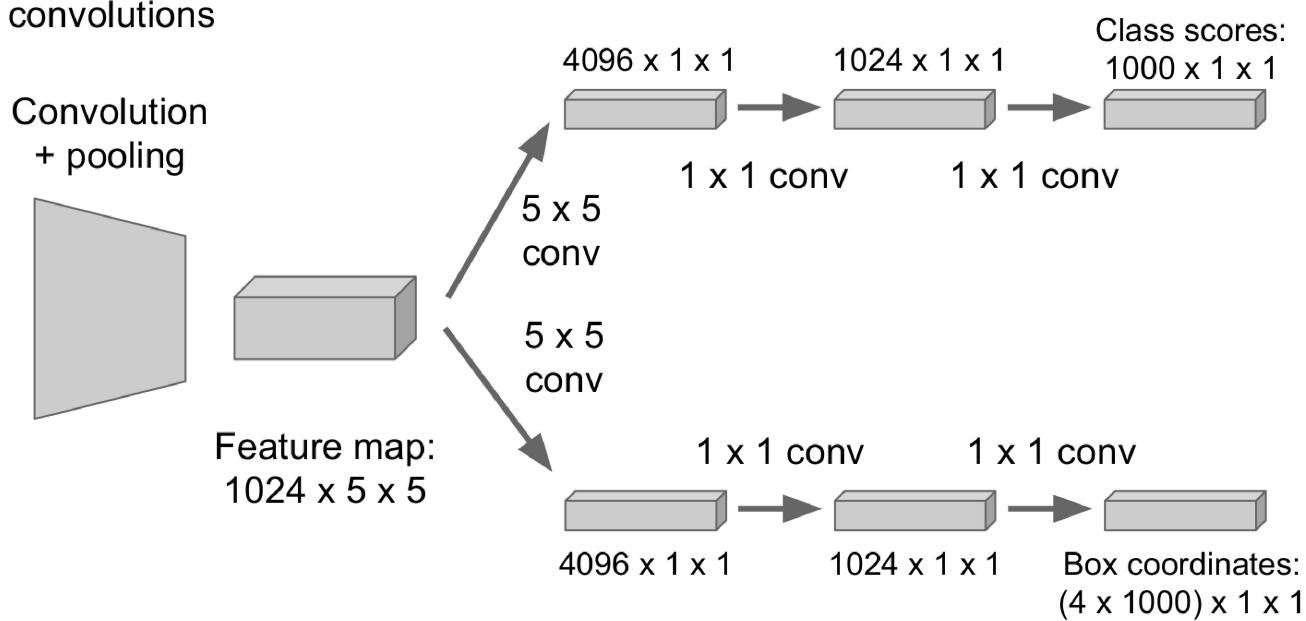


# Efficient Sliding Window: Overfeat

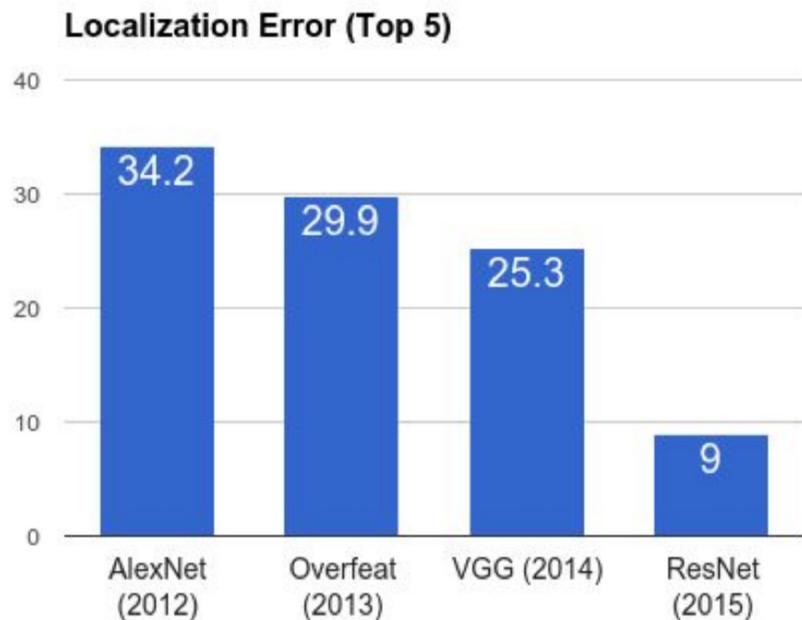
Efficient sliding window by converting fully-connected layers into convolutions



Image:  
 $3 \times 221 \times 221$



# ImageNet Classification + Localization



**AlexNet:** Localization method not published

**Overfeat:** Multiscale convolutional regression with box merging

**VGG:** Same as Overfeat, but fewer scales and locations; simpler method, gains all due to deeper features

**ResNet:** Different localization method (RPN) and much deeper features

# Object Detection

## Computer Vision Tasks

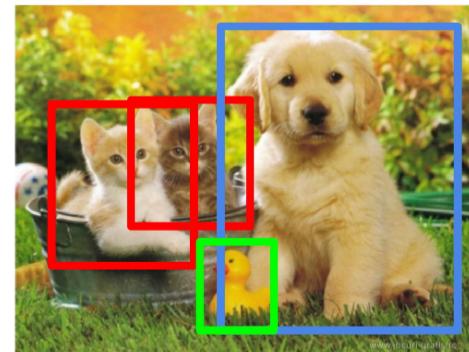
Classification



Classification  
+ Localization



Object Detection



Instance  
Segmentation



# Detection as Regression

## 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**

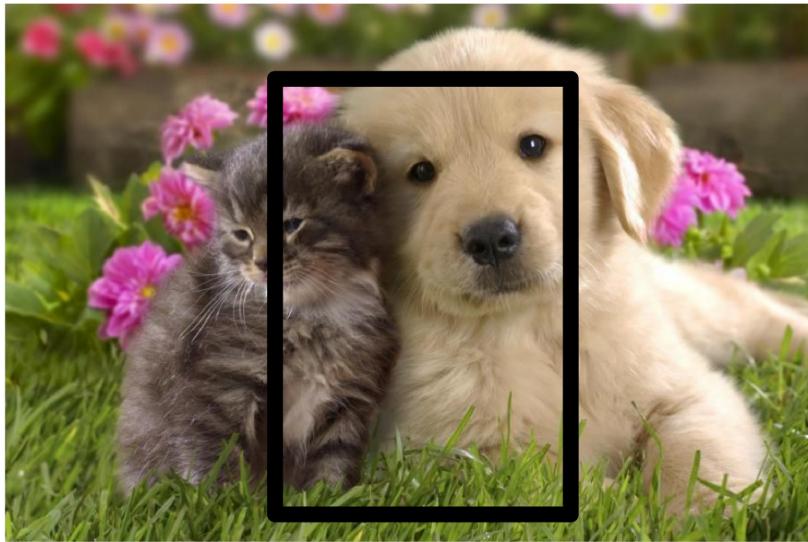
# Detection as Classification



**CAT? YES!**

**DOG? NO**

# Detection as Classification



**CAT? NO**

**DOG? NO**

## Detection as Classification

**Problem:** Need to test many positions and scales

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

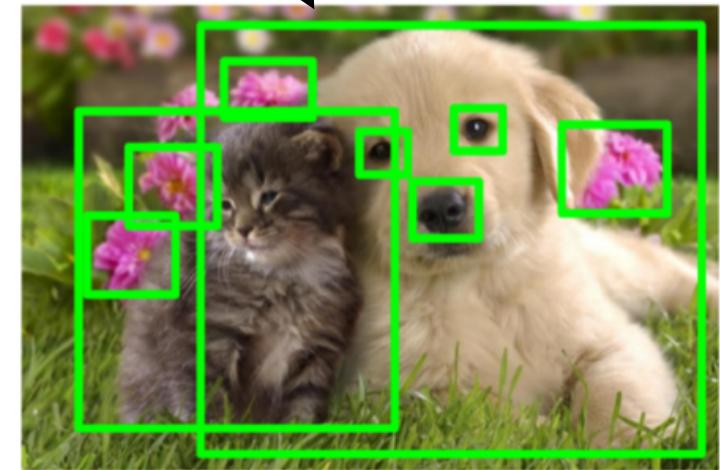
# Detection as Classification

**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

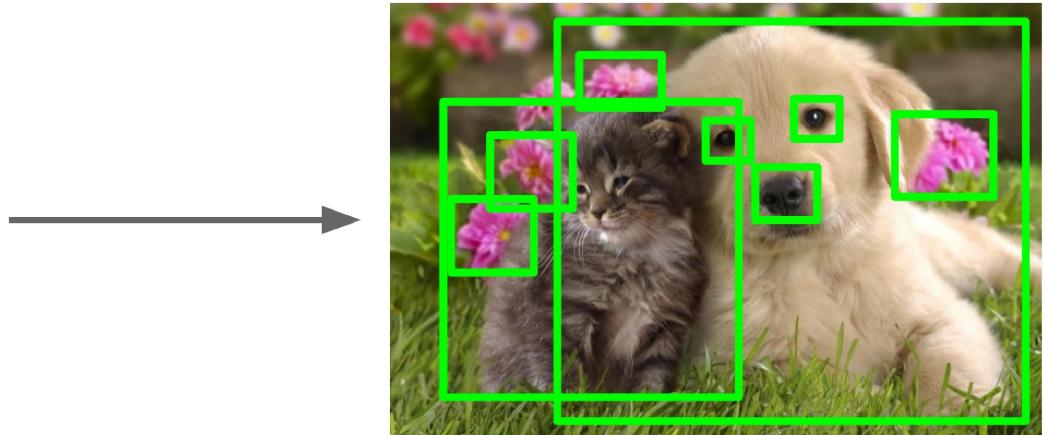
# Question?

- How to find the subset of proposals?



# Region Proposals

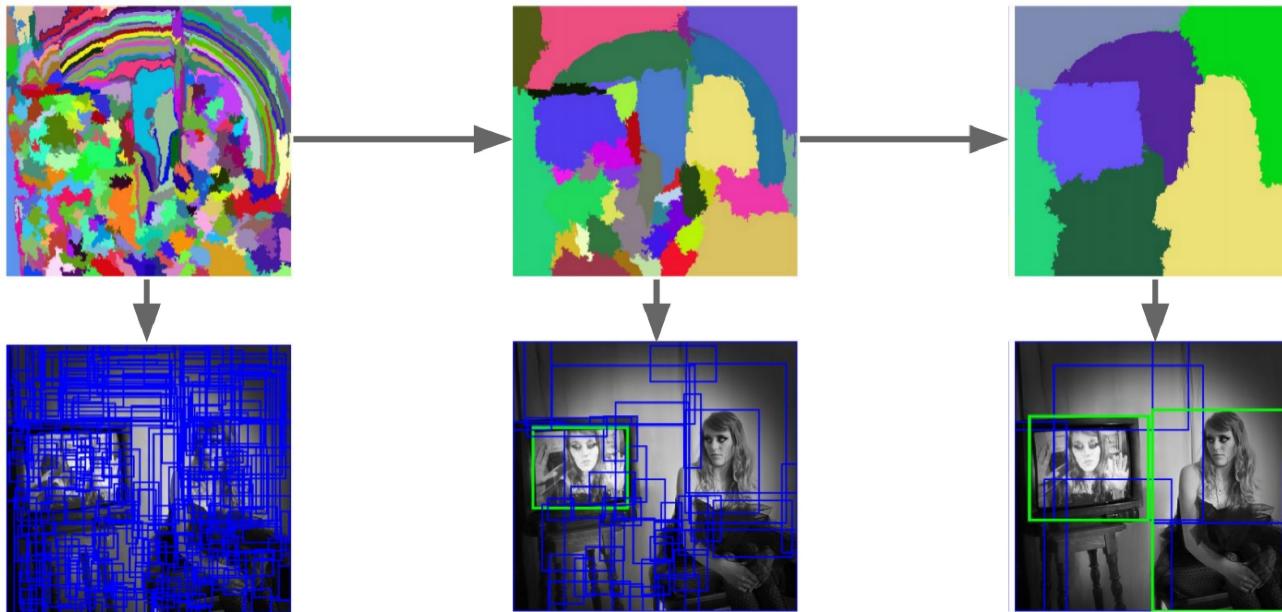
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



# Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert  
regions  
to boxes



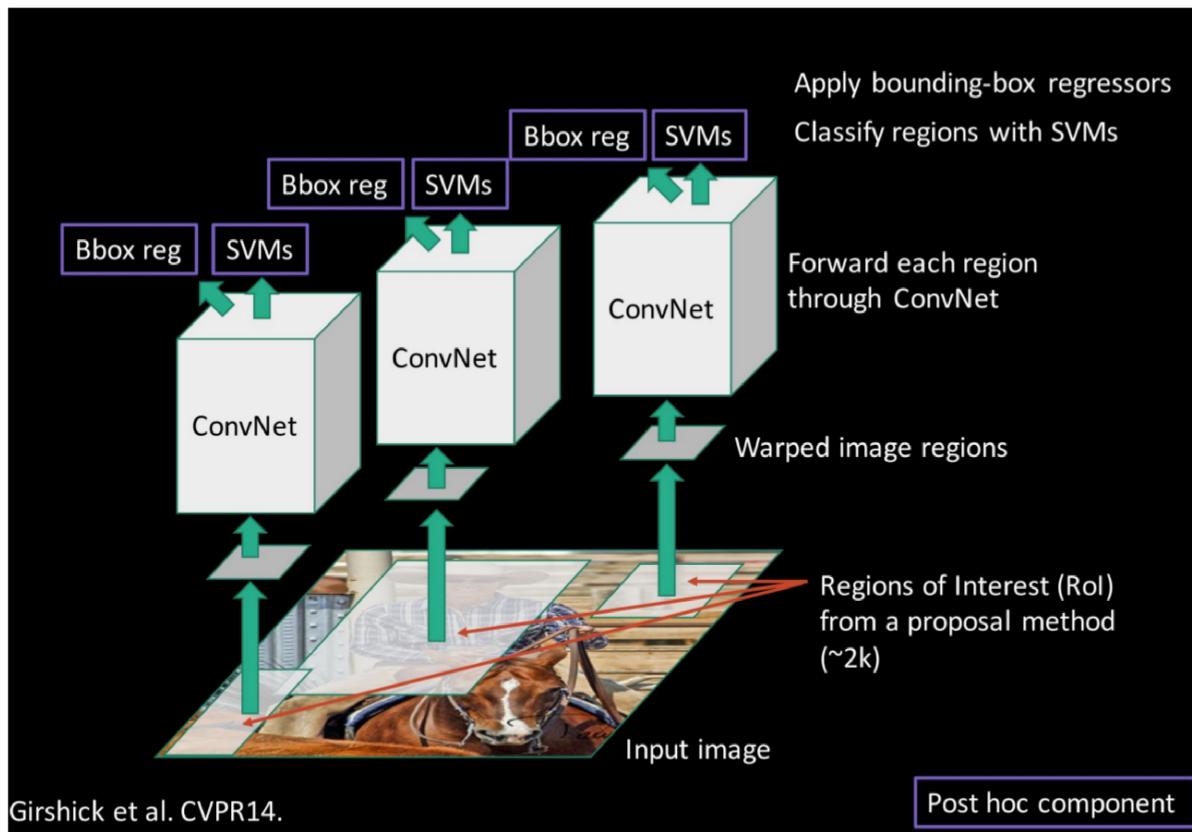
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013

# Region Proposals: Many other choices

Method	Approach	Outputs Segments	Outputs Score	Control #proposals	Time (sec.)	Repeatability	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

# Detection Algorithm: R-CNN

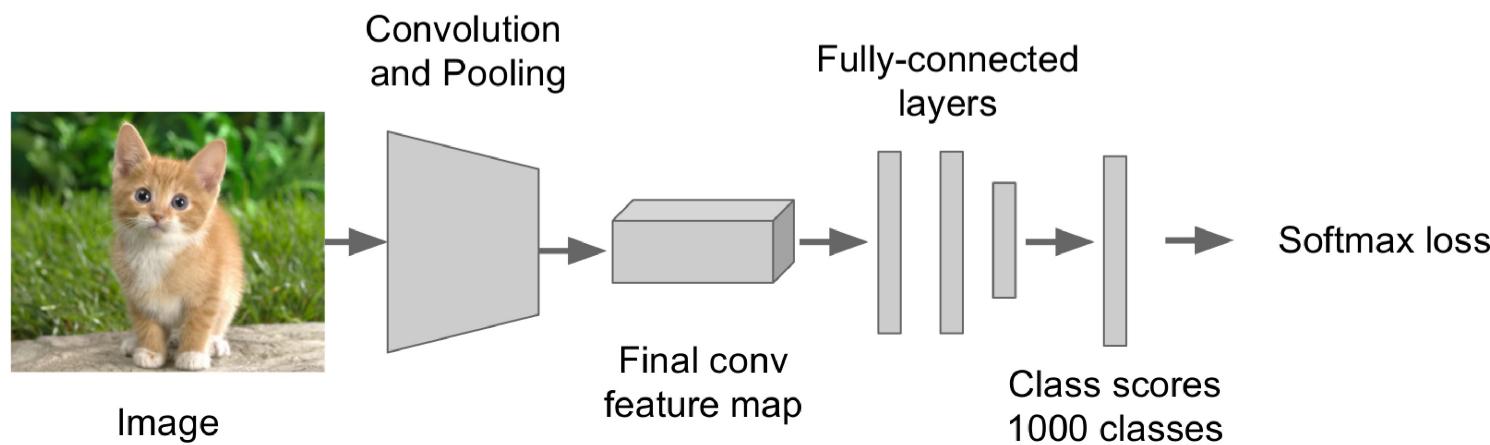


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

Slide credit: Ross Girshick

# R-CNN Training

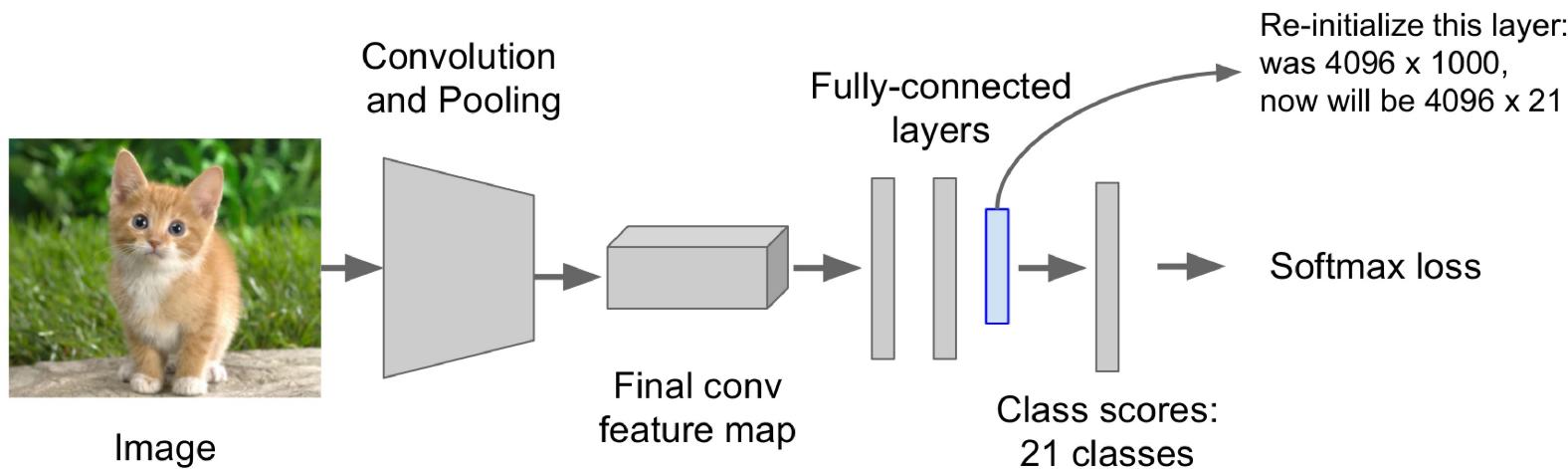
**Step 1:** Train (or download) a classification model for ImageNet (AlexNet)



# 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



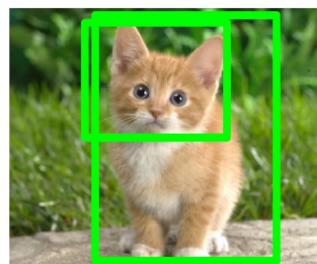
# R-CNN Training

## 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!



Image

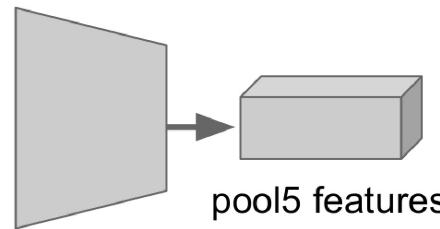


Region Proposals

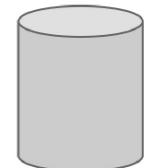


Crop + Warp

Convolution  
and Pooling



Forward pass



Save to disk

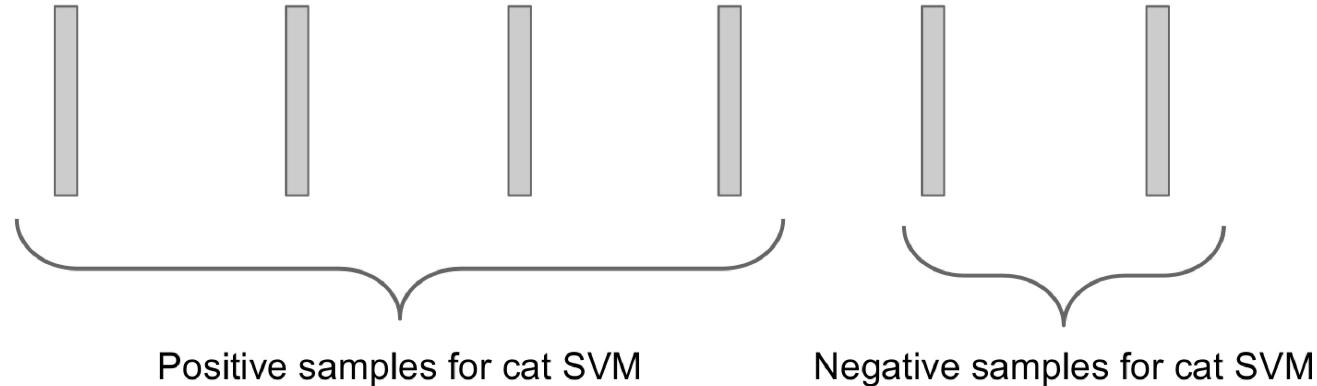
# R-CNN Training

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

Training image regions



Cached region features



Positive samples for cat SVM

Negative samples for cat SVM

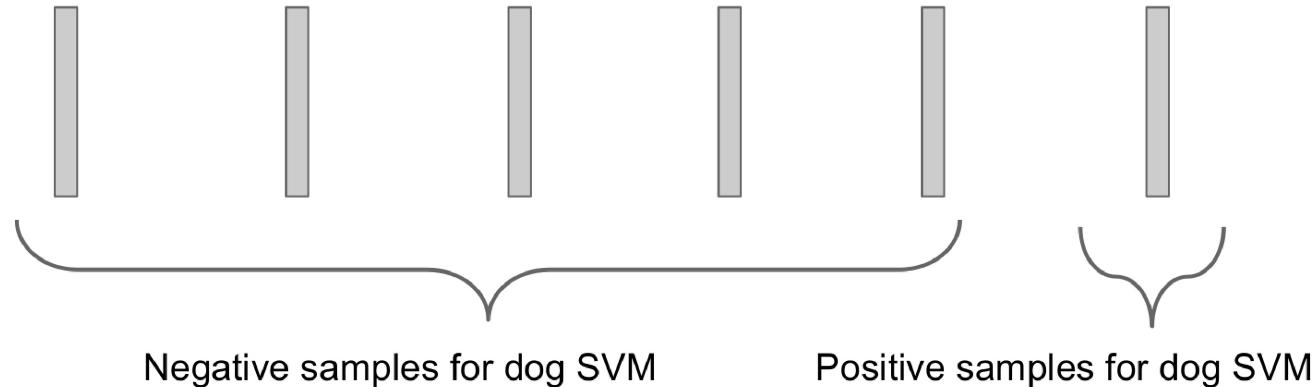
# R-CNN Training

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

Training image regions



Cached 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

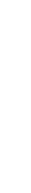


Regression targets  
( $dx$ ,  $dy$ ,  $dw$ ,  $dh$ )  
Normalized coordinates

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



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



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

# Object Detection: Datasets

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

# Object Detection: Evaluation

We use a metric called “mean average precision” (mAP)

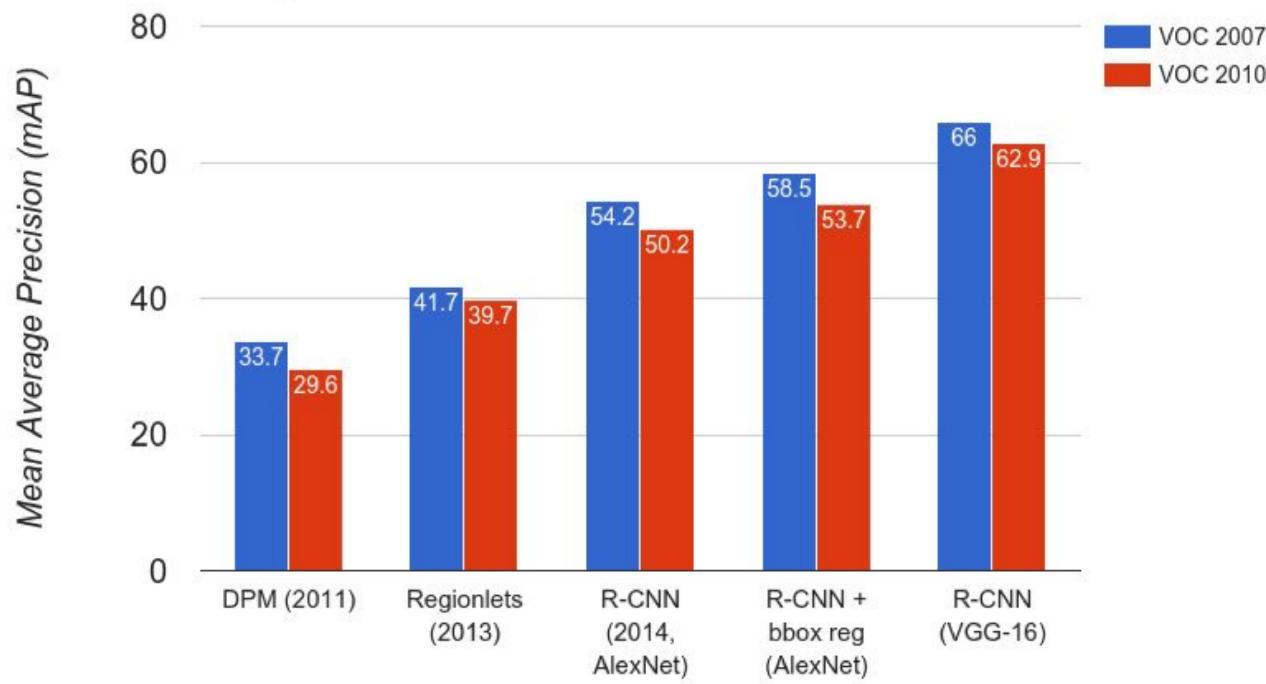
Compute average precision (AP) separately for each class, then average over classes

A detection is a true positive if it has IoU with a ground-truth box greater than some threshold (usually 0.5) (mAP@0.5)

Combine all detections from all test images to draw a precision / recall curve for each class; AP is area under the curve

TL;DR mAP is a number from 0 to 100; high is good

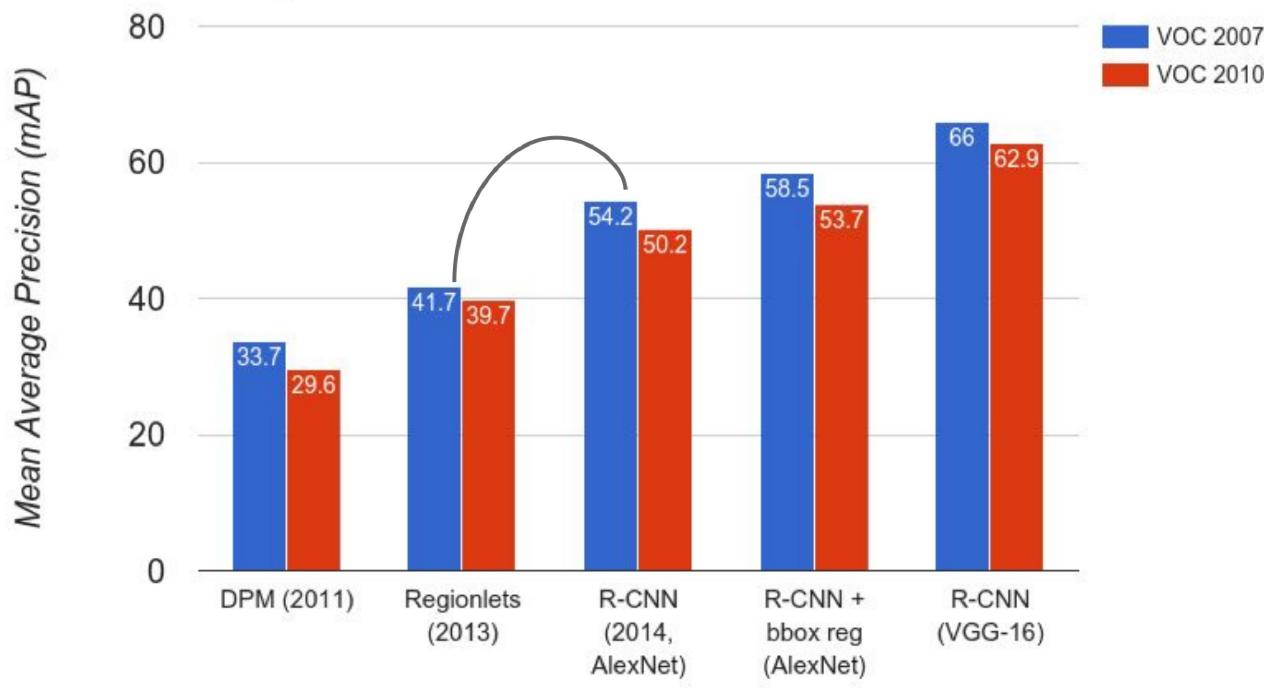
# R-CNN Results



Wang et al, "Regionlets for Generic Object Detection", ICCV 2013

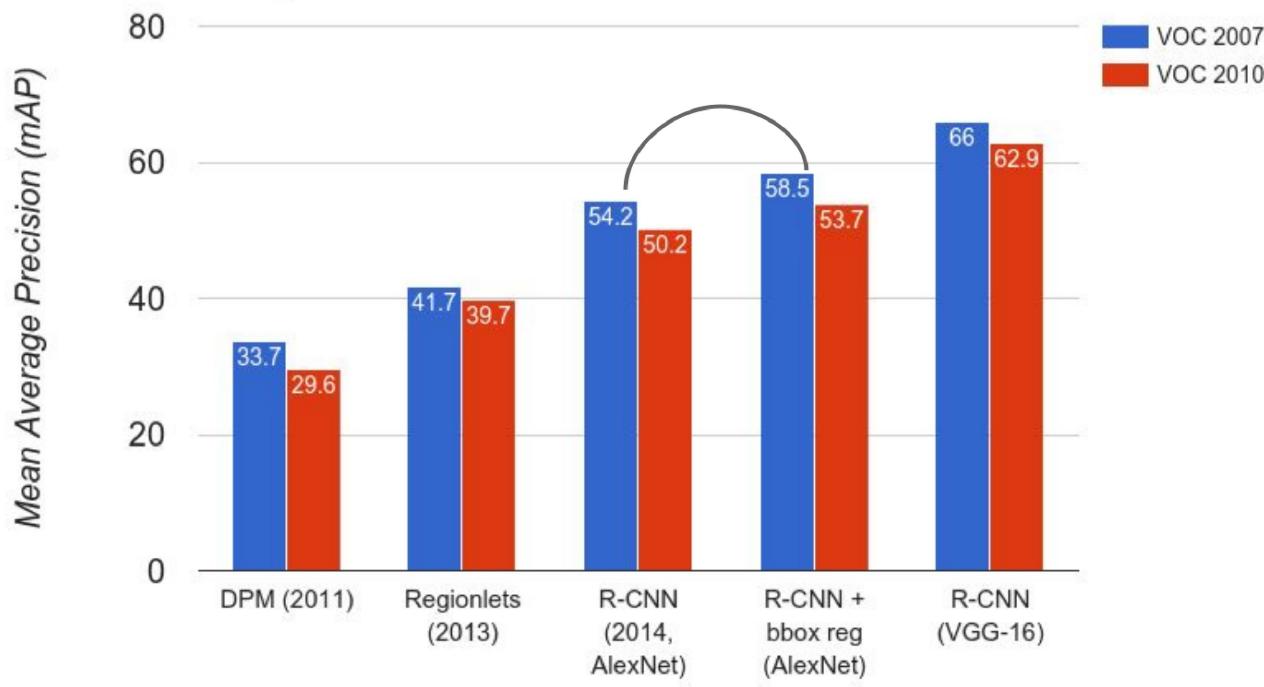
# R-CNN Results

Big improvement compared  
to pre-CNN methods



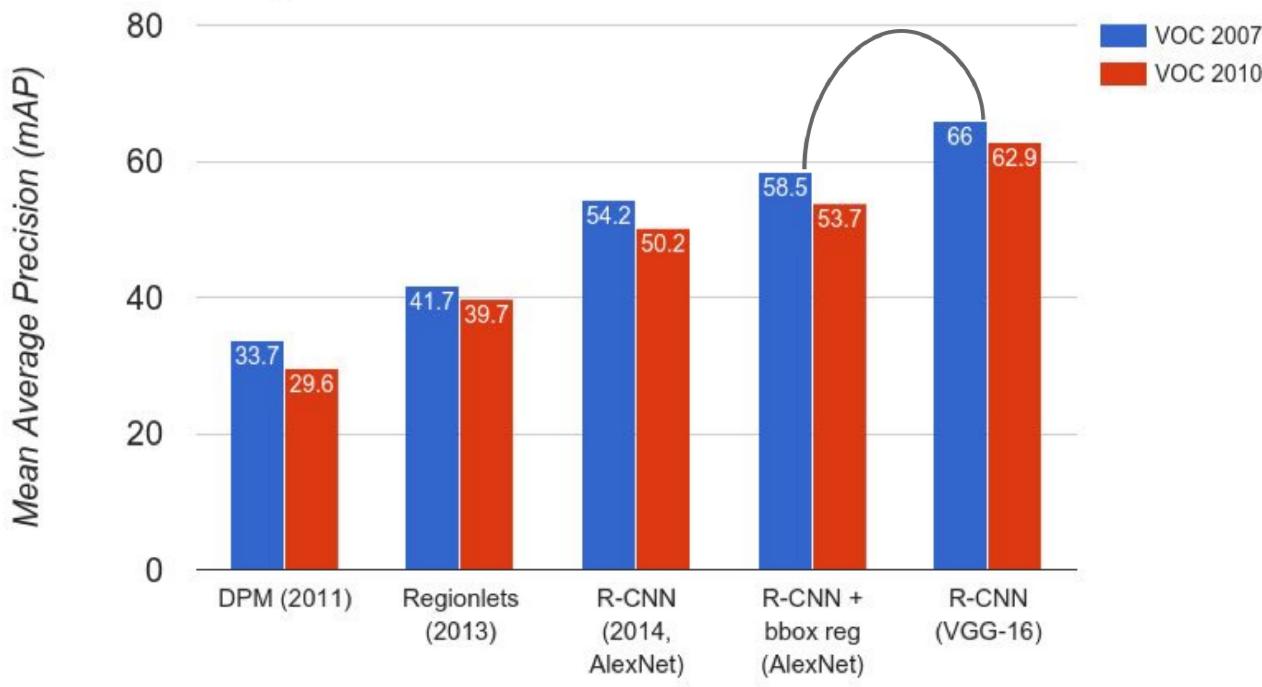
# R-CNN Results

Bounding box regression  
helps a bit



# R-CNN Results

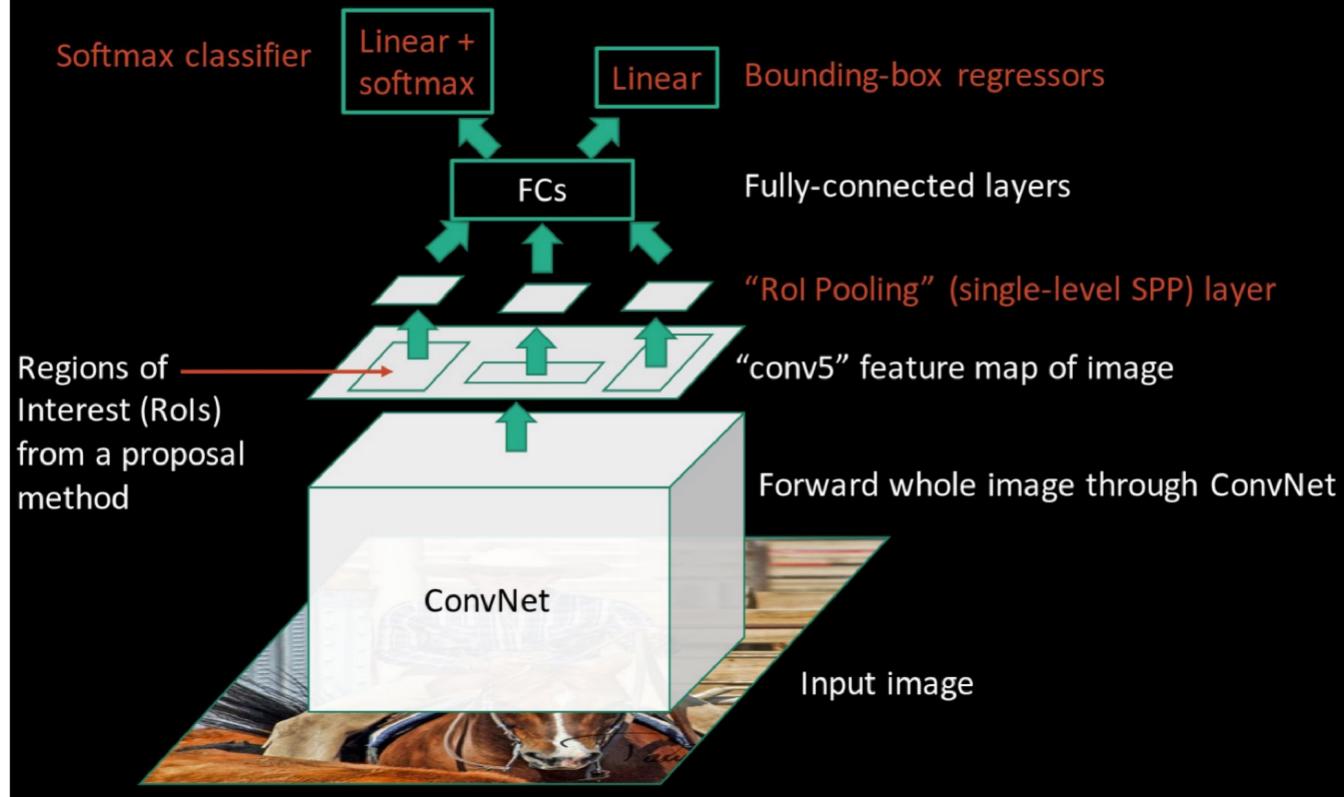
Features from a deeper network help a lot



## R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal
2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors
3. Complex multistage training pipeline

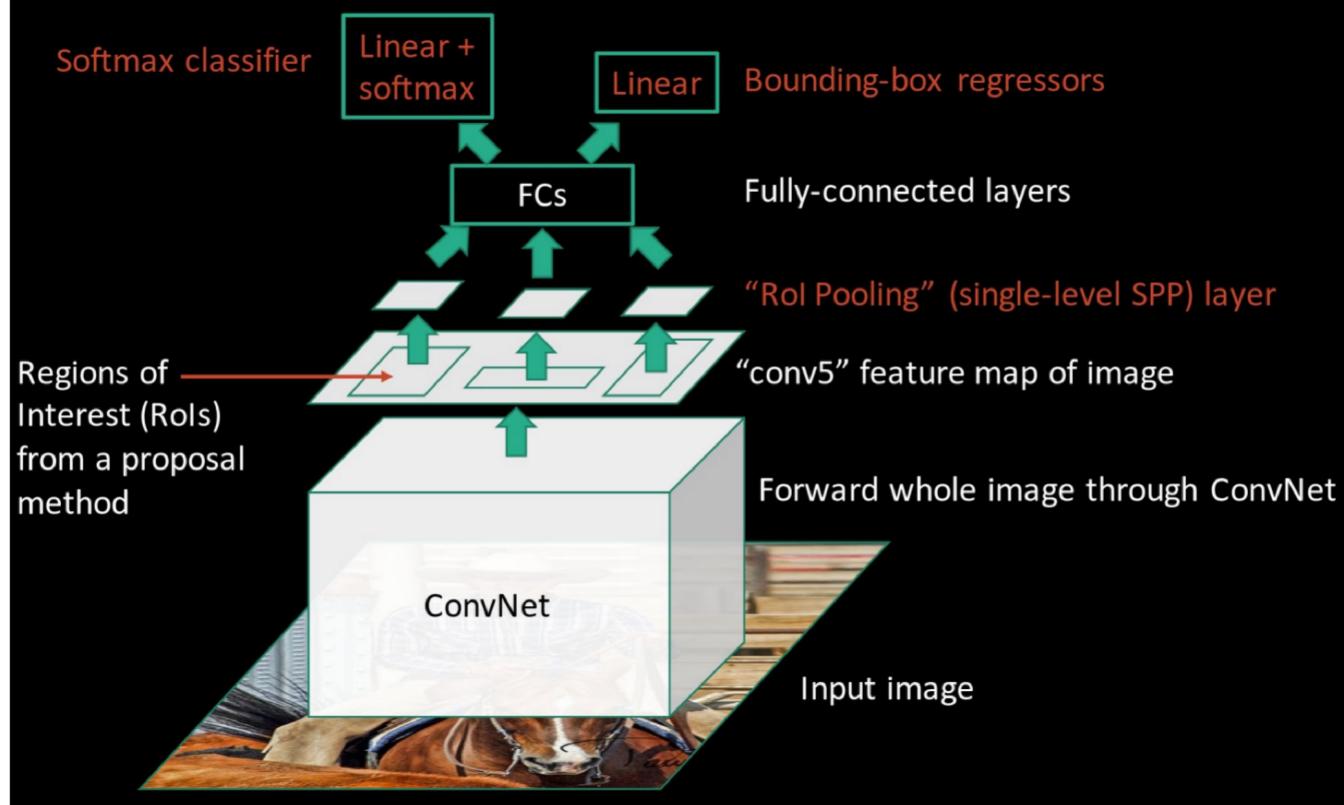
# Fast R-CNN (test time)



Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

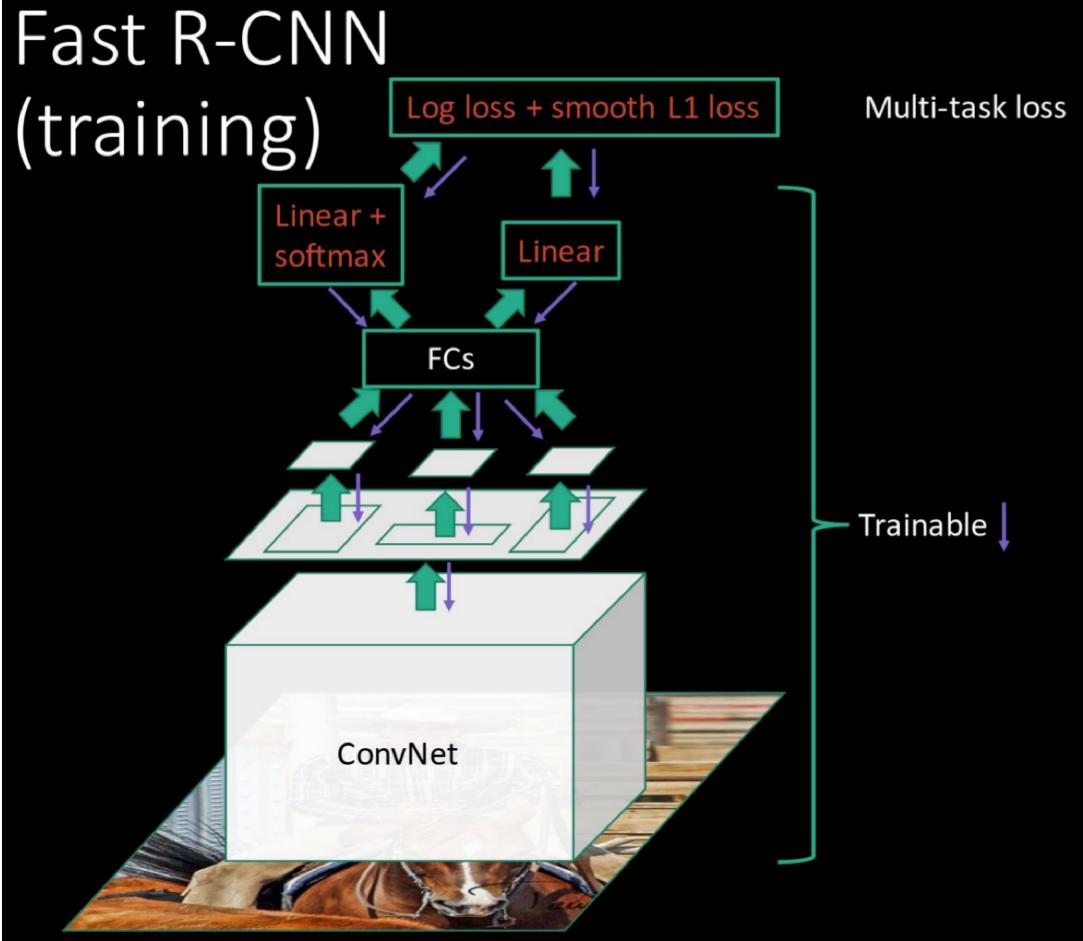
# Fast R-CNN (test time)



**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

# Fast R-CNN (training)



## 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!

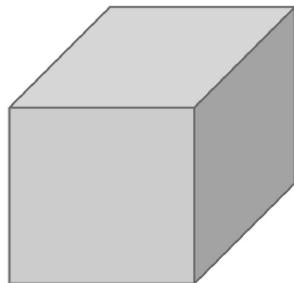
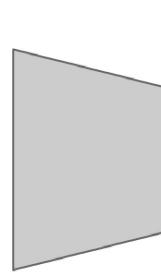
Slide credit: Ross Girshick

# Fast R-CNN: Region of Interest Pooling

Convolution  
and Pooling

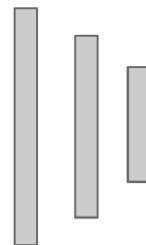


Hi-res input image:  
 $3 \times 800 \times 600$   
with region  
proposal



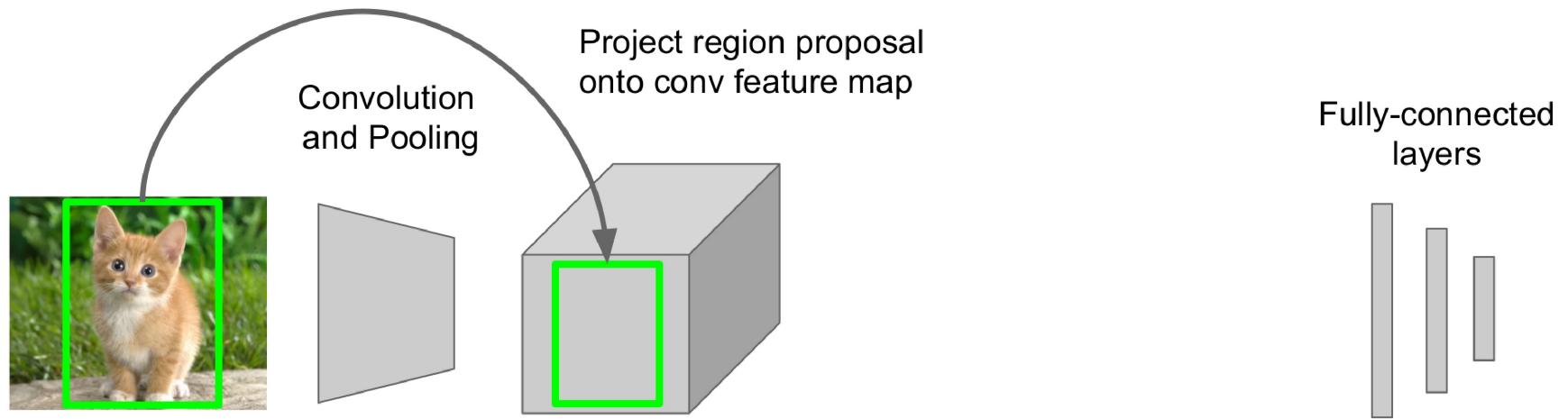
Hi-res conv features:  
 $C \times H \times W$   
with region proposal

Fully-connected  
layers



**Problem:** Fully-connected  
layers expect low-res conv  
features:  $C \times h \times w$

# Fast R-CNN: Region of Interest Pooling

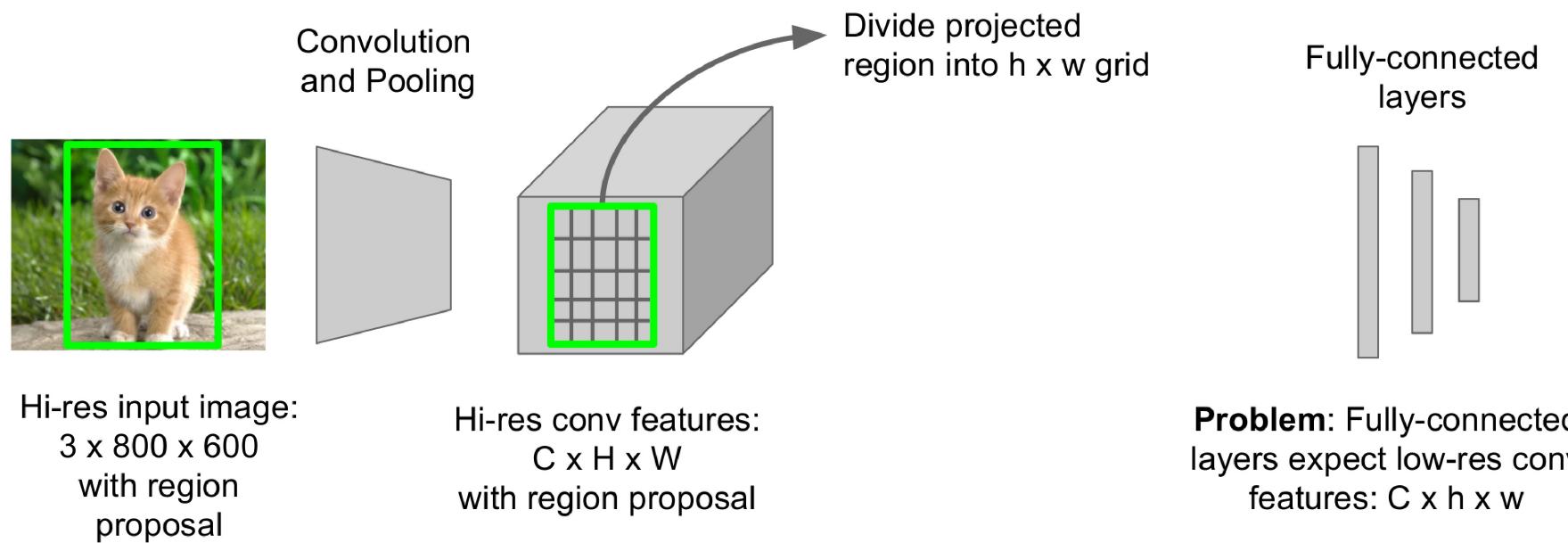


Hi-res input image:  
 $3 \times 800 \times 600$   
with region  
proposal

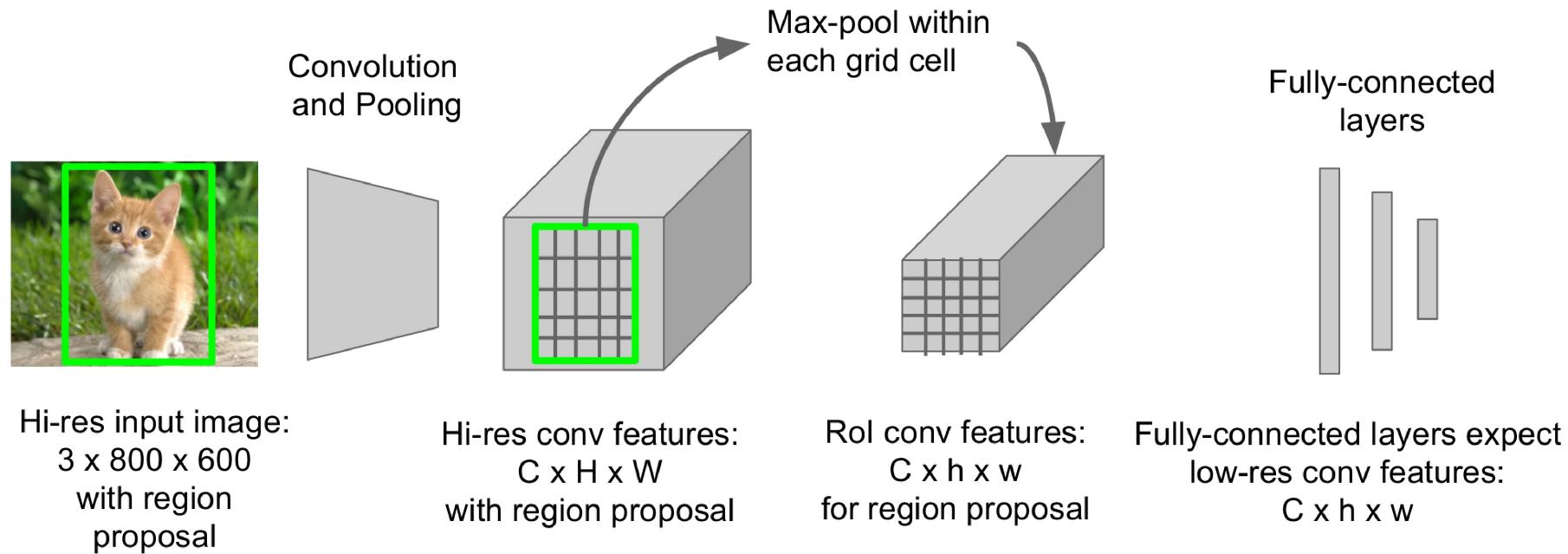
Hi-res conv features:  
 $C \times H \times W$   
with region proposal

**Problem:** Fully-connected  
layers expect low-res conv  
features:  $C \times h \times w$

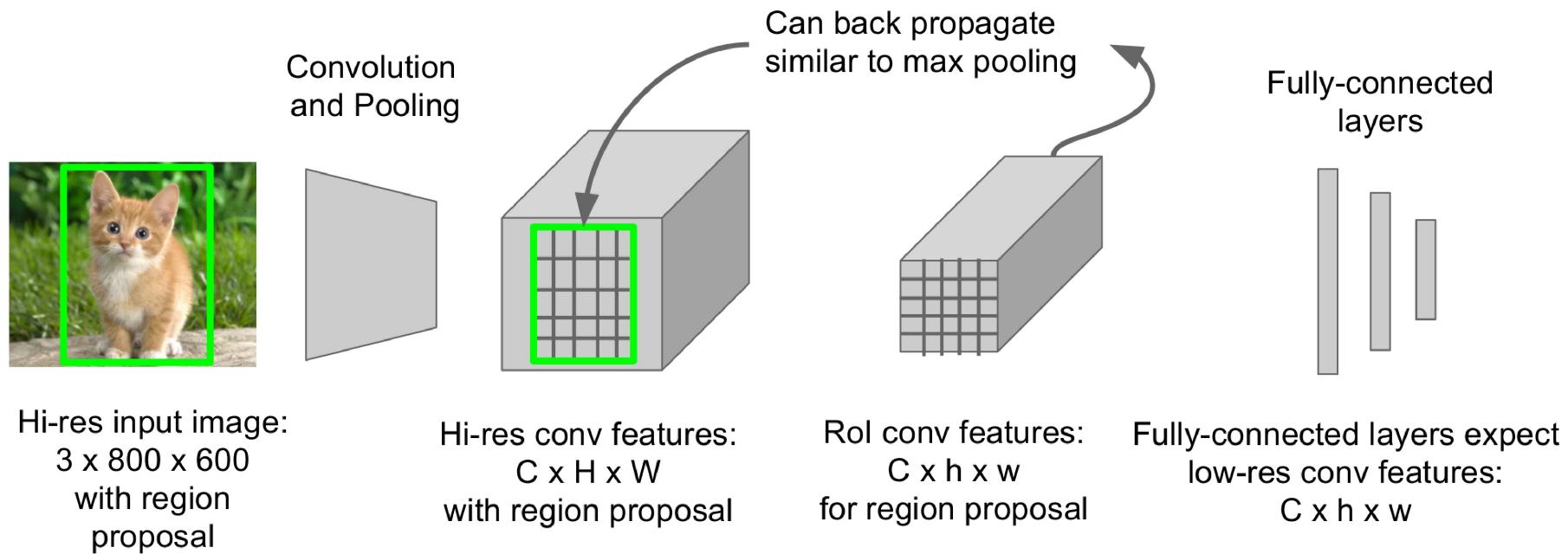
# Fast R-CNN: Region of Interest Pooling



# Fast R-CNN: Region of Interest Pooling



# Fast R-CNN: Region of Interest Pooling



# Fast R-CNN Results

Faster!

	R-CNN	Fast R-CNN
Training Time:	84 hours	<b>9.5 hours</b>
(Speedup)	1x	<b>8.8x</b>

Using VGG-16 CNN on Pascal VOC 2007 dataset

# Fast R-CNN Results

Faster!

FASTER!

	R-CNN	Fast R-CNN
Training Time:	84 hours	<b>9.5 hours</b>
(Speedup)	1x	<b>8.8x</b>
Test time per image	47 seconds	<b>0.32 seconds</b>
(Speedup)	1x	<b>146x</b>

Using VGG-16 CNN on Pascal VOC 2007 dataset

# Fast R-CNN Results

Faster!

FASTER!

Better!

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

Using VGG-16 CNN on Pascal VOC 2007 dataset

## Fast R-CNN Problem:

Test-time speeds don't include region proposals

	R-CNN	Fast R-CNN
Test time per image	47 seconds	<b>0.32 seconds</b>
(Speedup)	1x	<b>146x</b>
Test time per image with Selective Search	50 seconds	<b>2 seconds</b>
(Speedup)	1x	<b>25x</b>

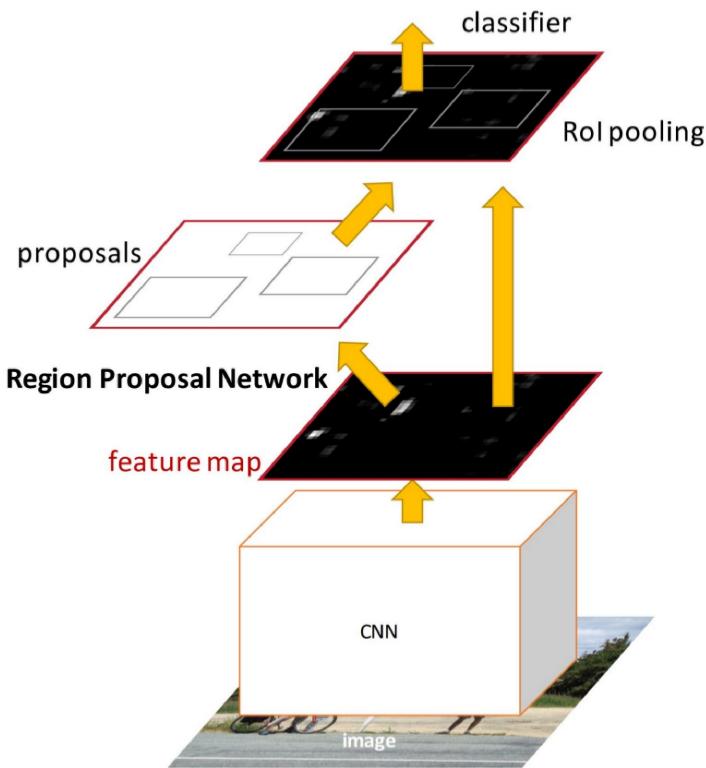
# Fast R-CNN Problem Solution:

Test-time speeds don't include region proposals

Just make the CNN do region proposals too!

	R-CNN	Fast R-CNN
Test time per image	47 seconds	<b>0.32 seconds</b>
(Speedup)	1x	<b>146x</b>
Test time per image with Selective Search	50 seconds	<b>2 seconds</b>
(Speedup)	1x	<b>25x</b>

# Faster R-CNN:



Insert a **Region Proposal Network (RPN)** after the last convolutional layer

RPN trained to produce region proposals directly; no need for external region proposals!

After RPN, use RoI Pooling and an upstream classifier and bbox regressor just like Fast R-CNN

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

Slide credit: Ross Girshick

# Faster R-CNN: Region Proposal Network

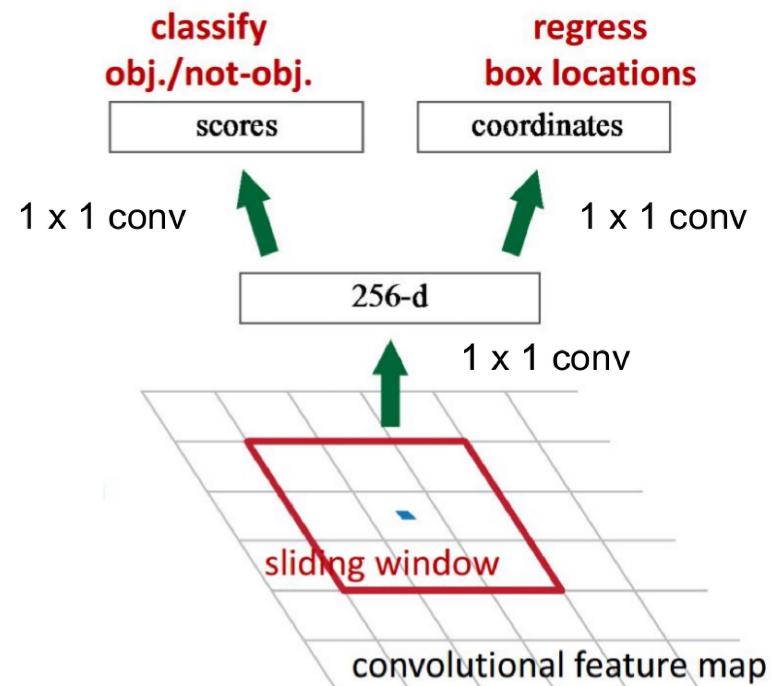
Slide a small window on the feature map

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



Slide credit: Kaiming He

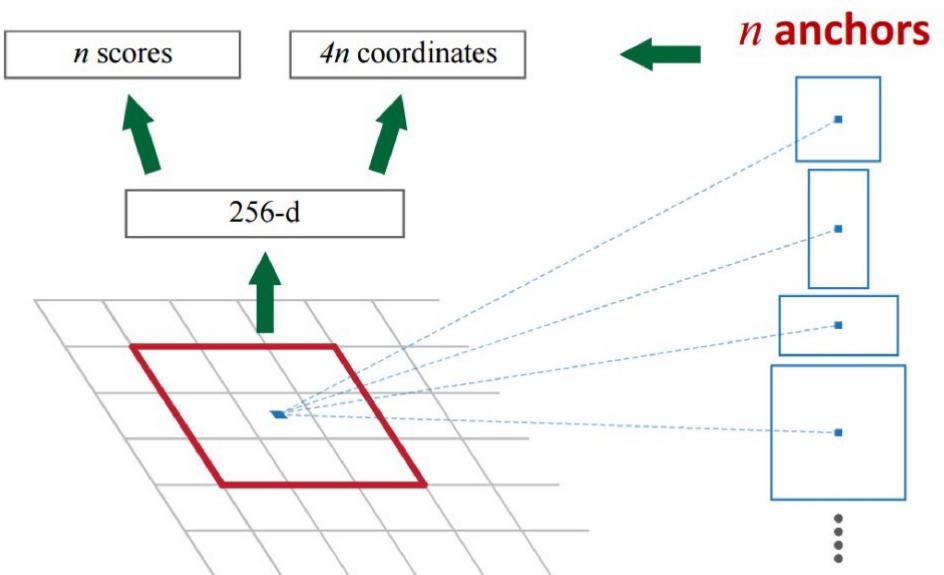
# Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

Anchors are **translation invariant**: use the same ones at every location

Regression gives offsets from anchor boxes

Classification gives the probability that each (regressed) anchor shows an object



# Faster R-CNN: Training

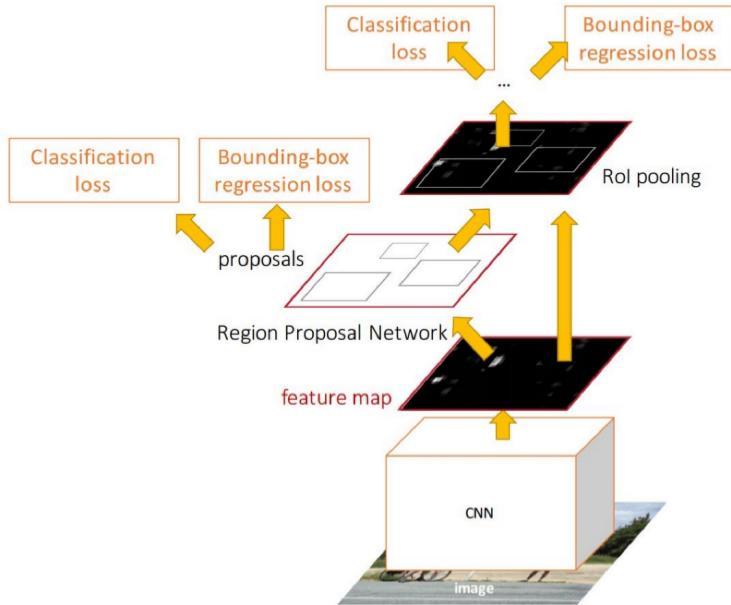
In the paper: Ugly pipeline

- 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  $\rightarrow$  proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal  $\rightarrow$  box)



Slide credit: Ross Girshick

# Faster R-CNN: Results

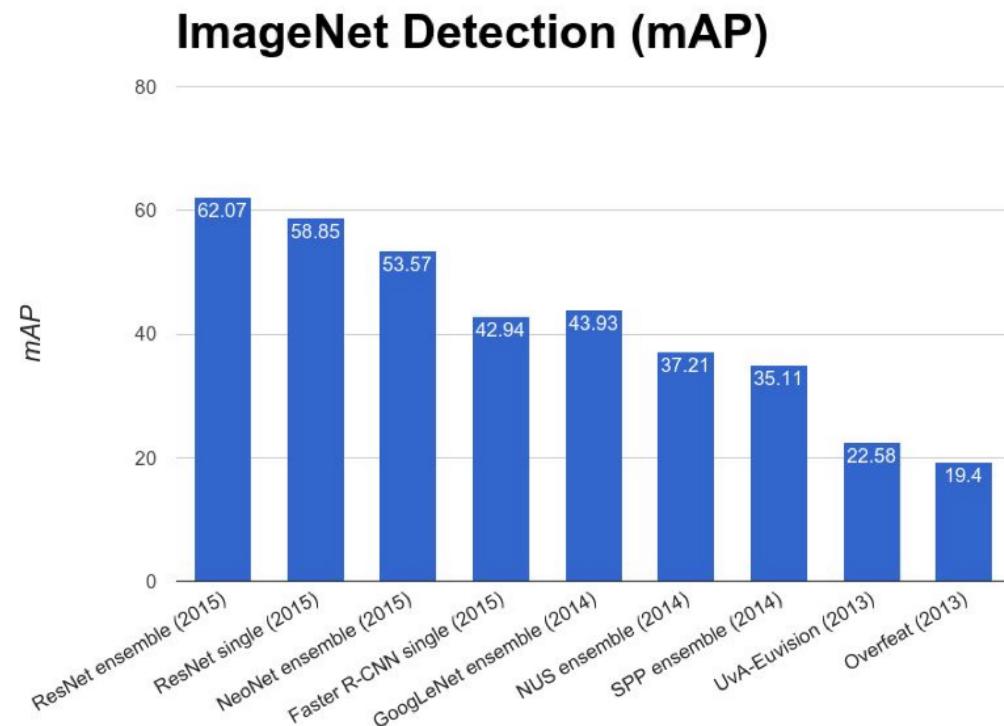
	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	<b>0.2 seconds</b>
(Speedup)	1x	25x	<b>250x</b>
mAP (VOC 2007)	66.0	<b>66.9</b>	<b>66.9</b>

# Object Detection State-of-the-art: ResNet 101 + Faster R-CNN + some extras

training data	COCO train		COCO trainval	
test data	COCO val		COCO test-dev	
mAP	@ .5	@ [.5, .95]	@ .5	@ [.5, .95]
baseline Faster R-CNN (VGG-16)	41.5	21.2		
baseline Faster R-CNN (ResNet-101)	48.4	27.2		
+box refinement	49.9	29.9		
+context	51.1	30.0	53.3	32.2
+multi-scale testing	53.8	32.5	<b>55.7</b>	<b>34.9</b>
ensemble			<b>59.0</b>	<b>37.4</b>

He et. al, "Deep Residual Learning for Image Recognition", arXiv 2015

# ImageNet Detection 2013 - 2015



# Object Detection code links:

## R-CNN

(Caffe + 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](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?

# Recap

## **Localization:**

- Find a fixed number of objects (one or many)
- L2 regression from CNN features to box coordinates
- Much simpler than detection; consider it for your projects!
- Overfeat: Regression + efficient sliding window with FC -> conv conversion
- Deeper networks do better

## **Object Detection:**

- Find a variable number of objects by classifying image regions
- Before CNNs: dense multiscale sliding window (HoG, DPM)
- Avoid dense sliding window with region proposals
- R-CNN: Selective Search + CNN classification / regression
- Fast R-CNN: Swap order of convolutions and region extraction
- Faster R-CNN: Compute region proposals within the network
- Deeper networks do better