

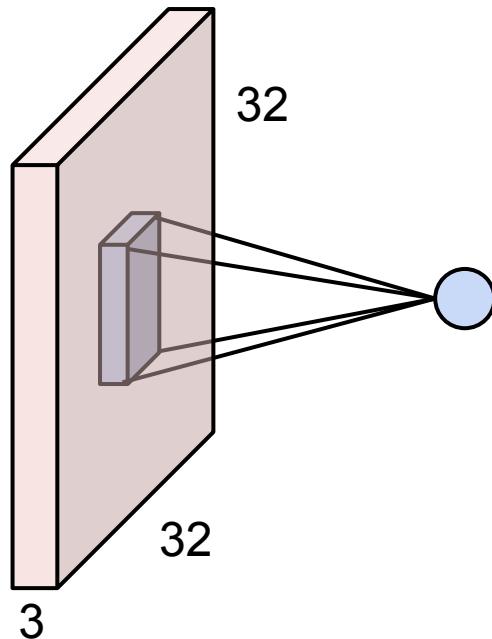
Lecture 8:

Spatial Localization and Detection

Administrative

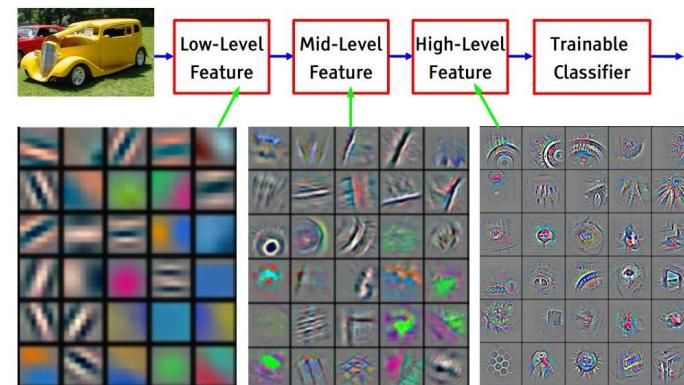
- Project Proposals were due on Saturday
- Homework 2 due Friday 2/5
- Homework 1 grades out this week
- Midterm will be in-class on Wednesday 2/10

Convolution



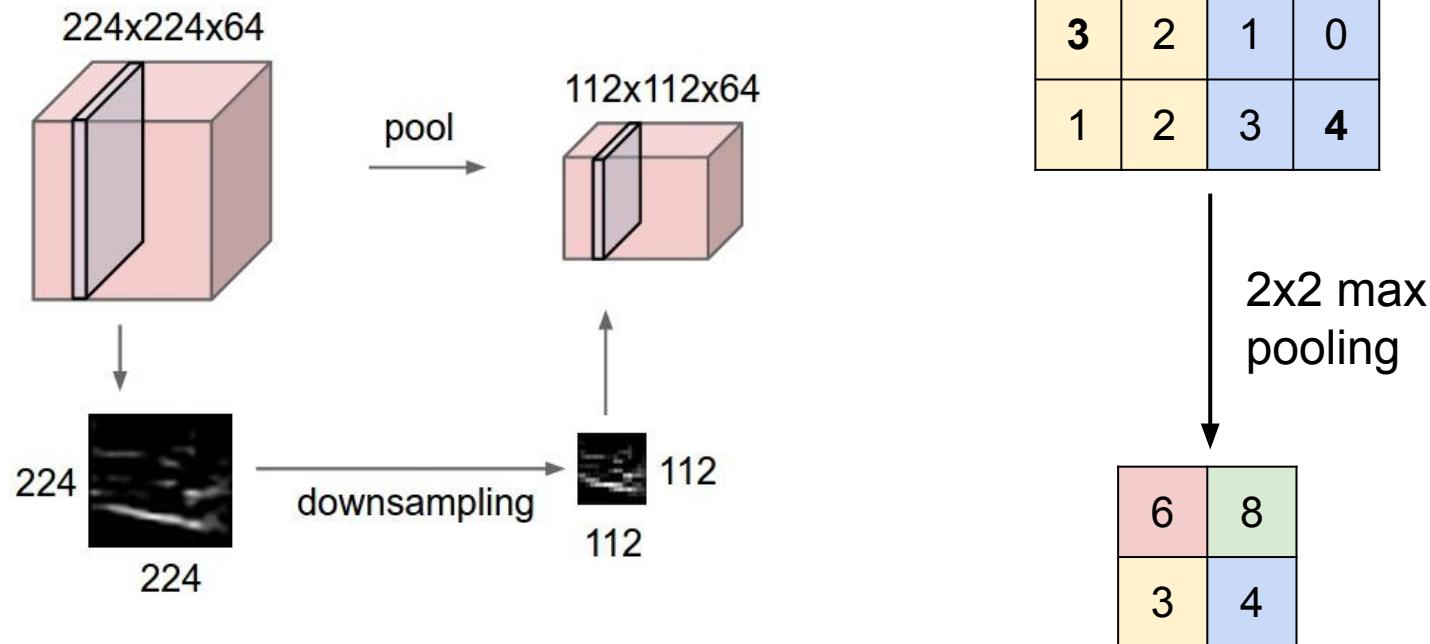
Summary. To summarize, the Conv Layer:

- 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 .



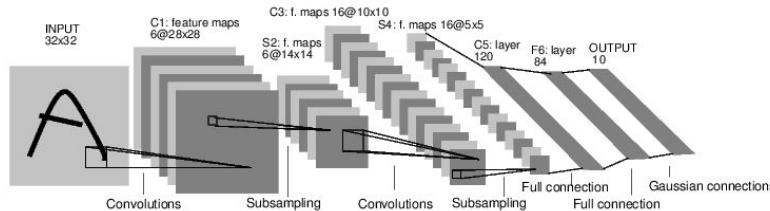
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Pooling

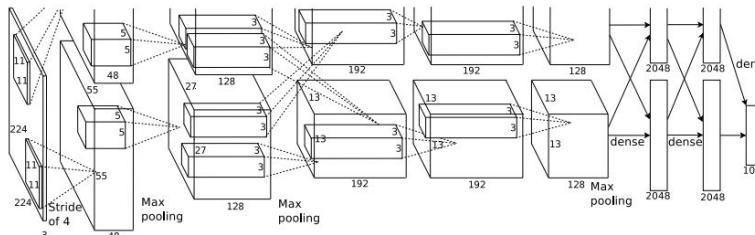


Case Studies

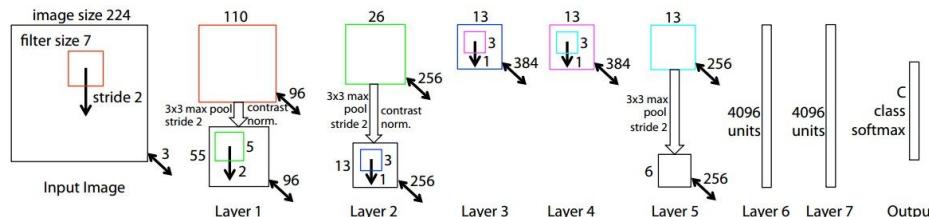
LeNet
(1998)



AlexNet
(2012)



ZFNet
(2013)



Case Studies

D	E
16 weight layers	19 weight layers
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
conv3-512	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	

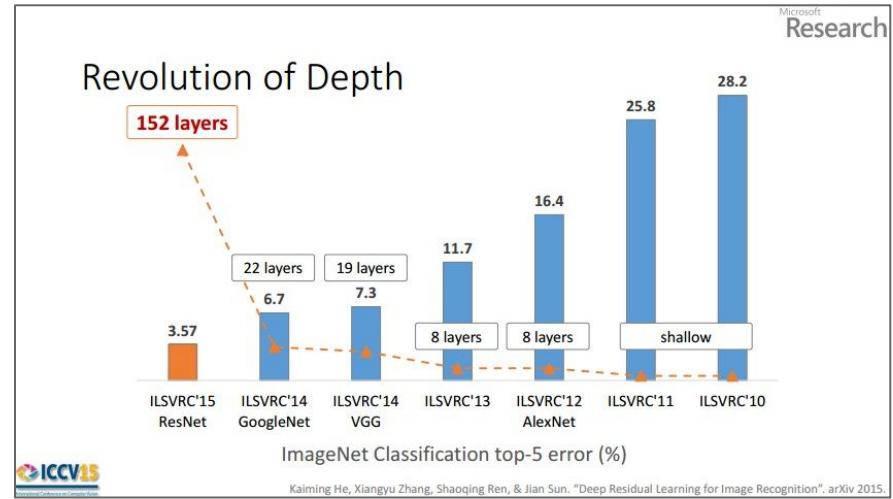


VGG
(2014)

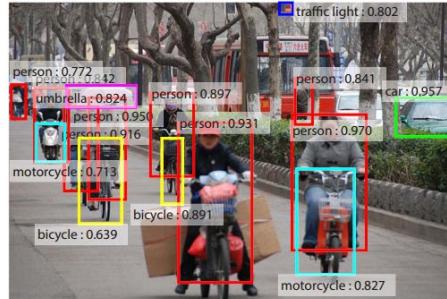
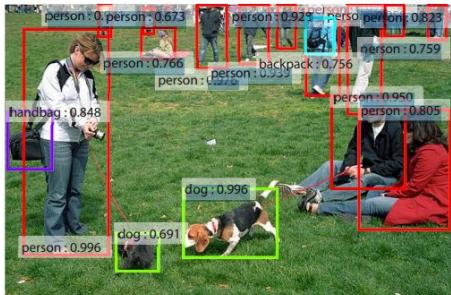
GoogLeNet
(2014)



ResNet
(2015)



Localization and Detection



Results from Faster R-CNN, Ren et al 2015

Computer Vision Tasks

Classification



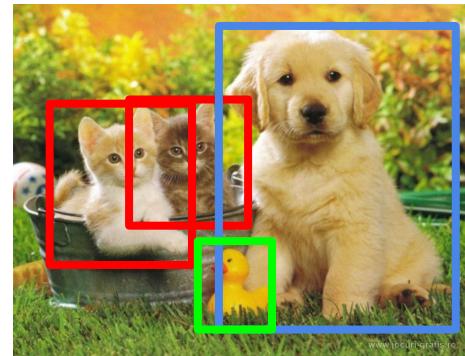
CAT

Classification + Localization



CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



CAT, DOG, DUCK

Single object

Multiple objects

Computer Vision Tasks

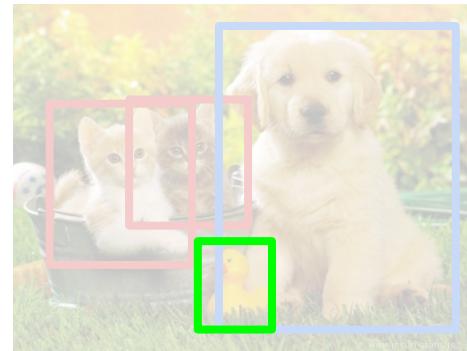
Classification



**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

Classification + Localization: ImageNet

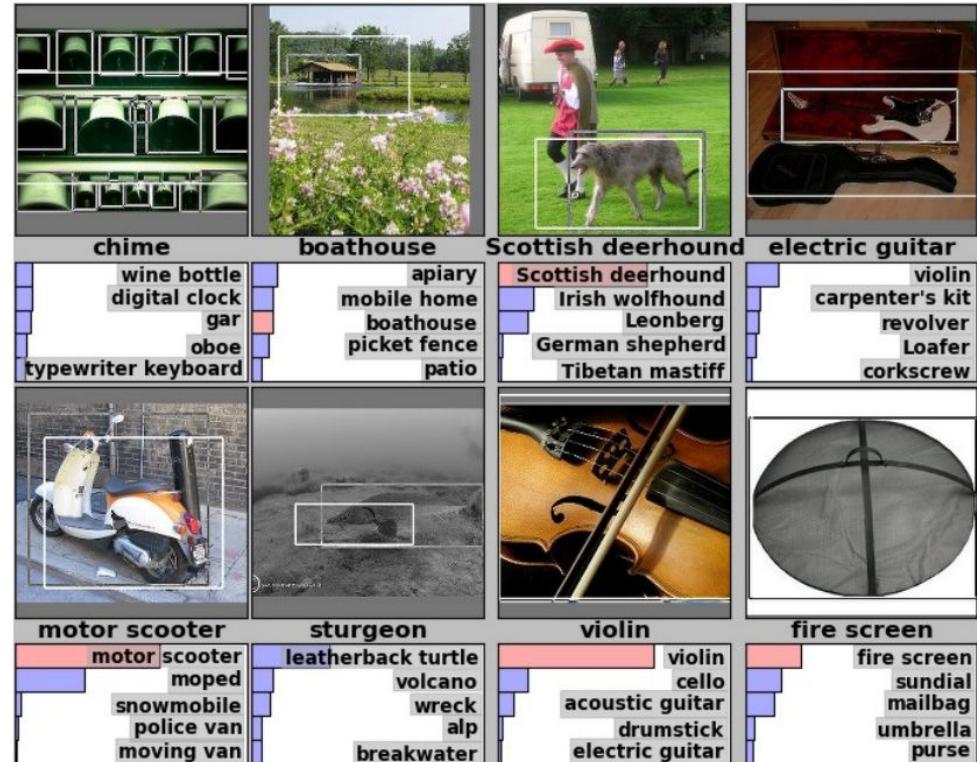
1000 classes (same as classification)

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

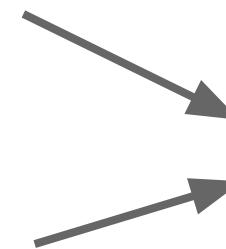
Idea #1: Localization as Regression

Input: image



Neural Net
→

Output:
Box coordinates
(4 numbers)



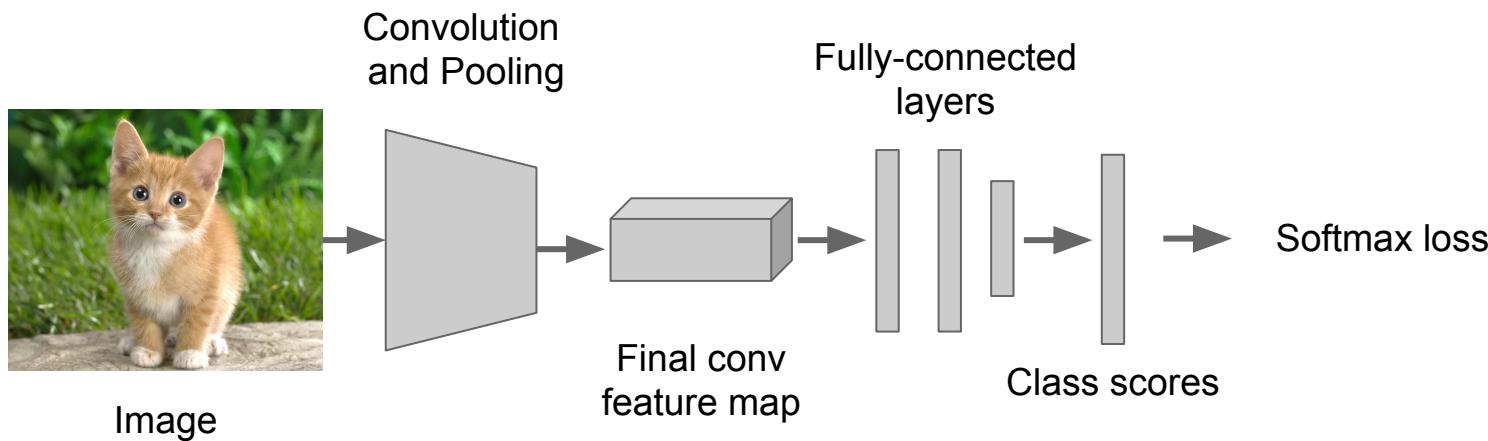
Loss:
L2 distance

Correct output:
box coordinates
(4 numbers)

Only one object,
simpler than detection

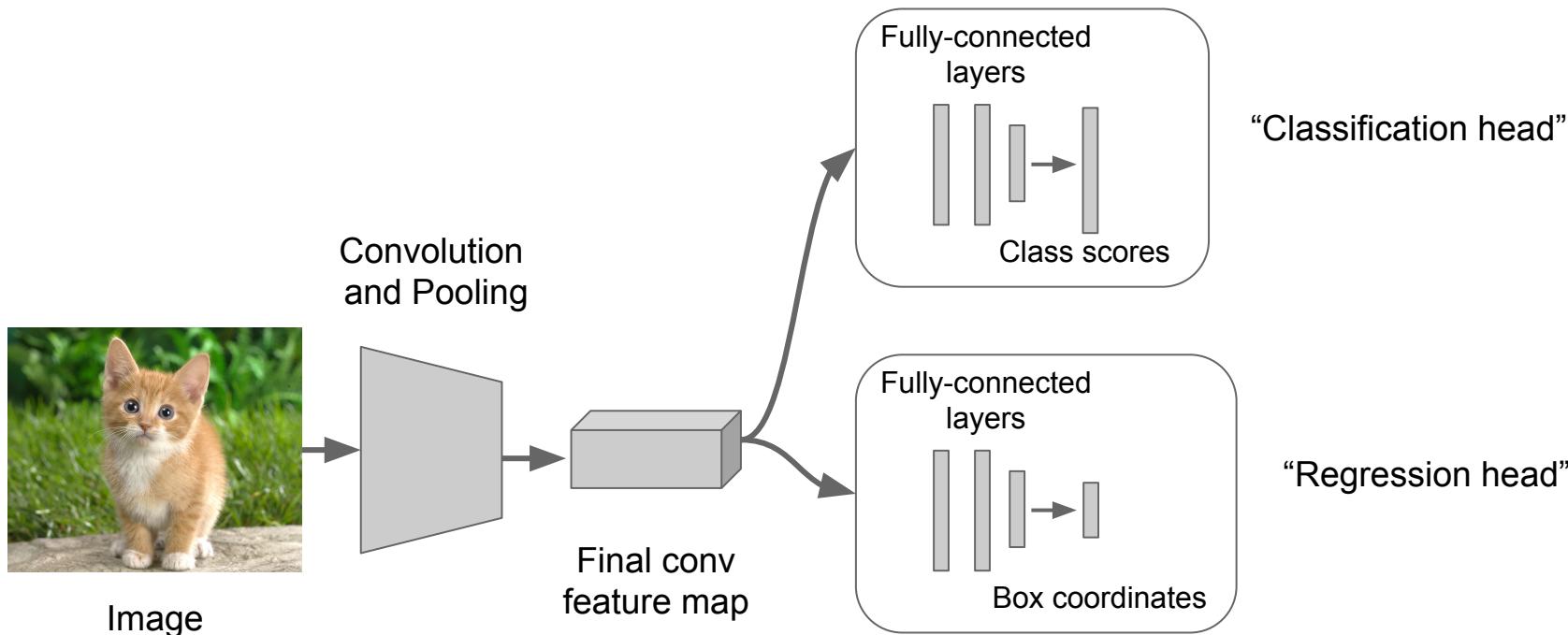
Simple Recipe for Classification + Localization

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



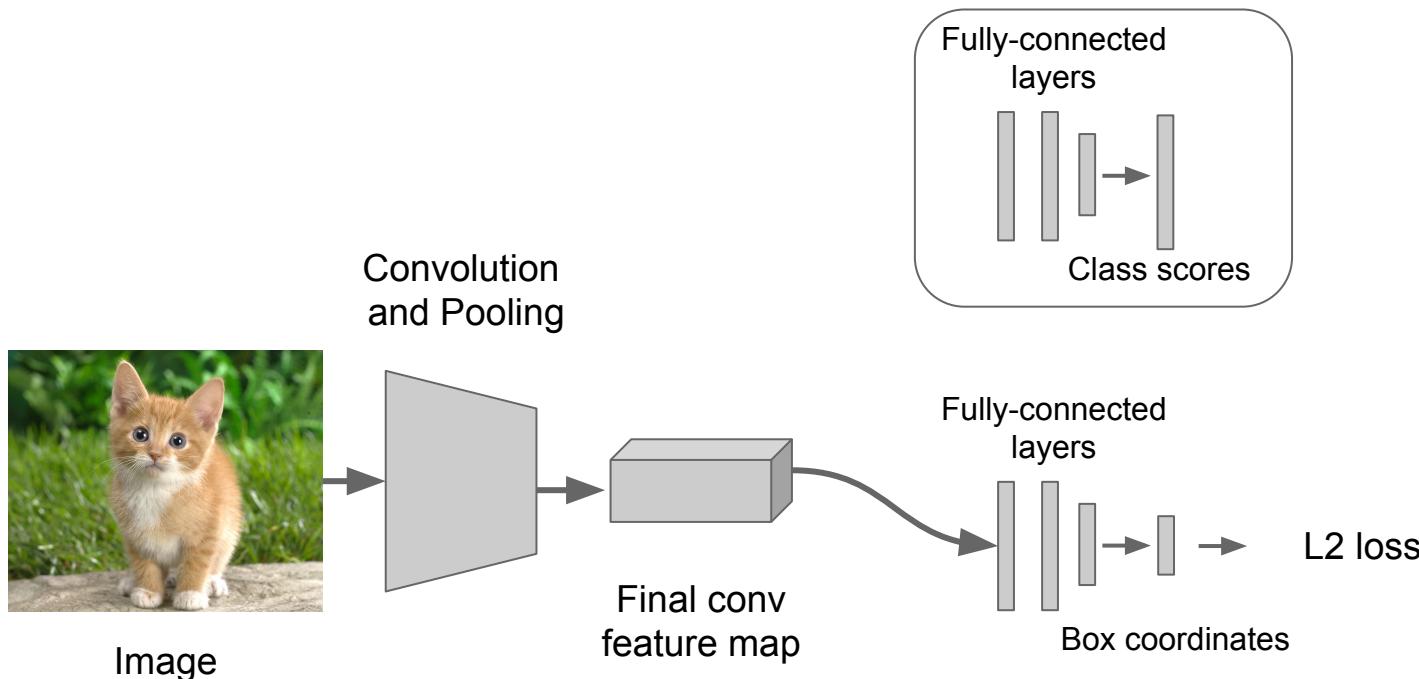
Simple Recipe for Classification + Localization

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



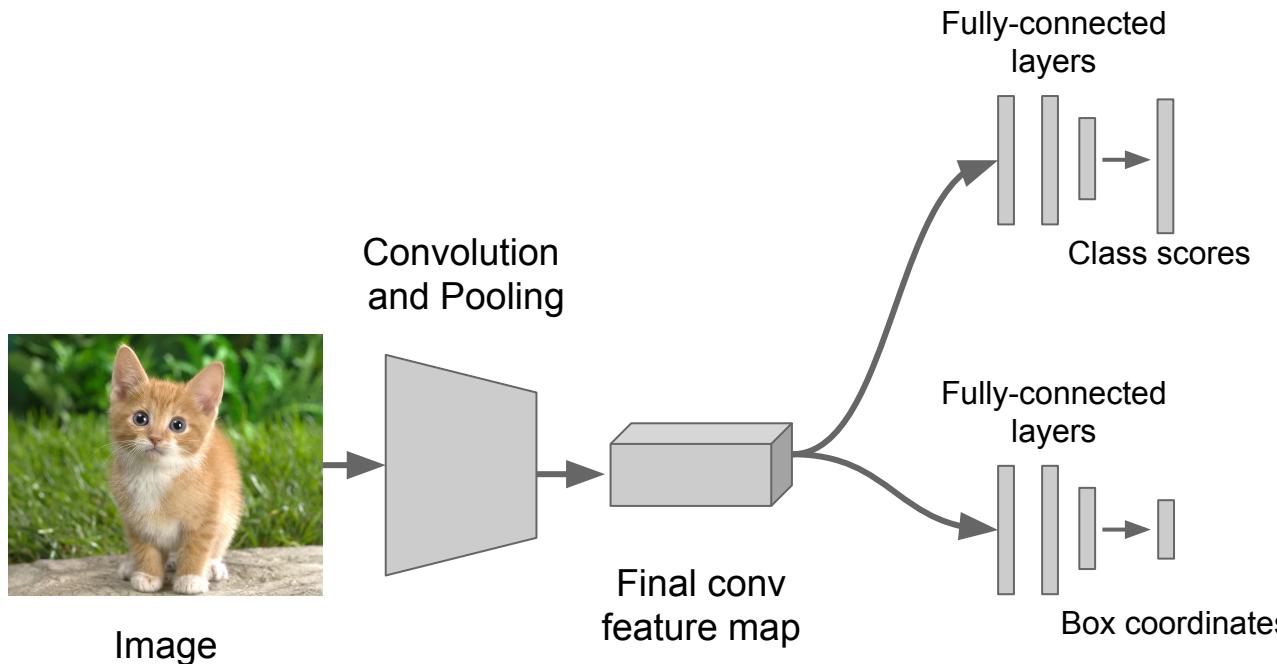
Simple Recipe for Classification + Localization

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



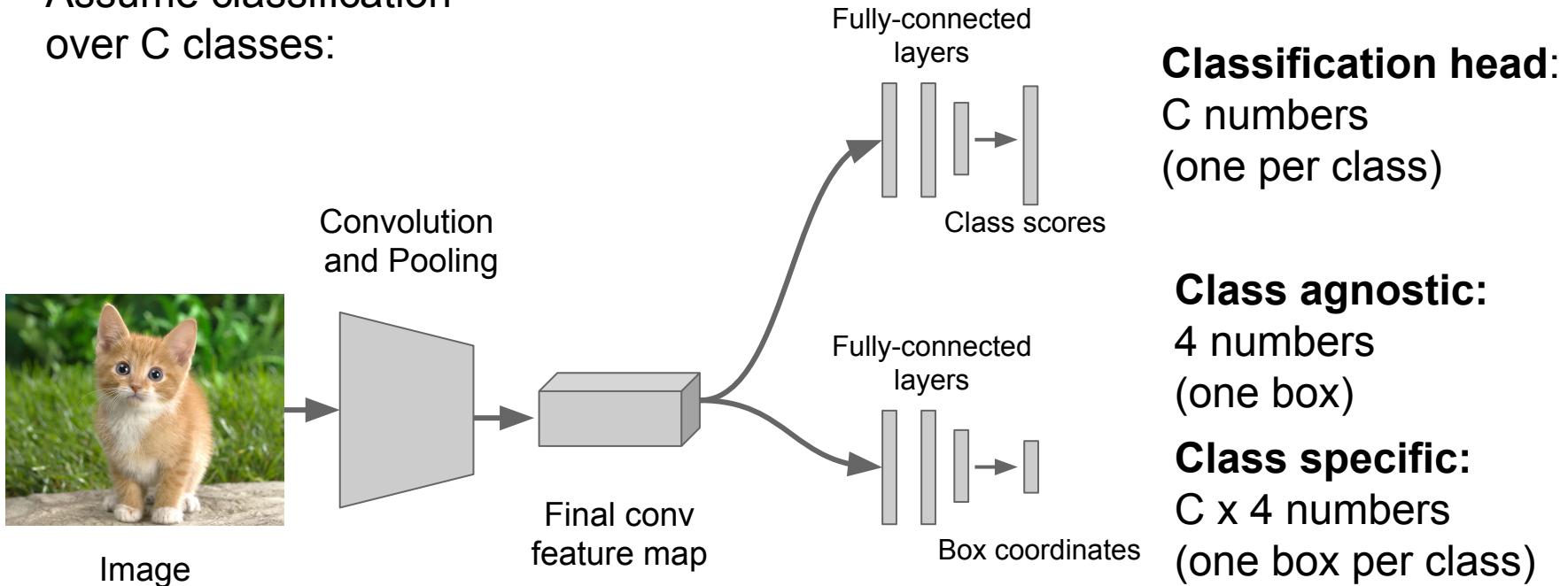
Simple Recipe for Classification + Localization

Step 4: At test time use both heads

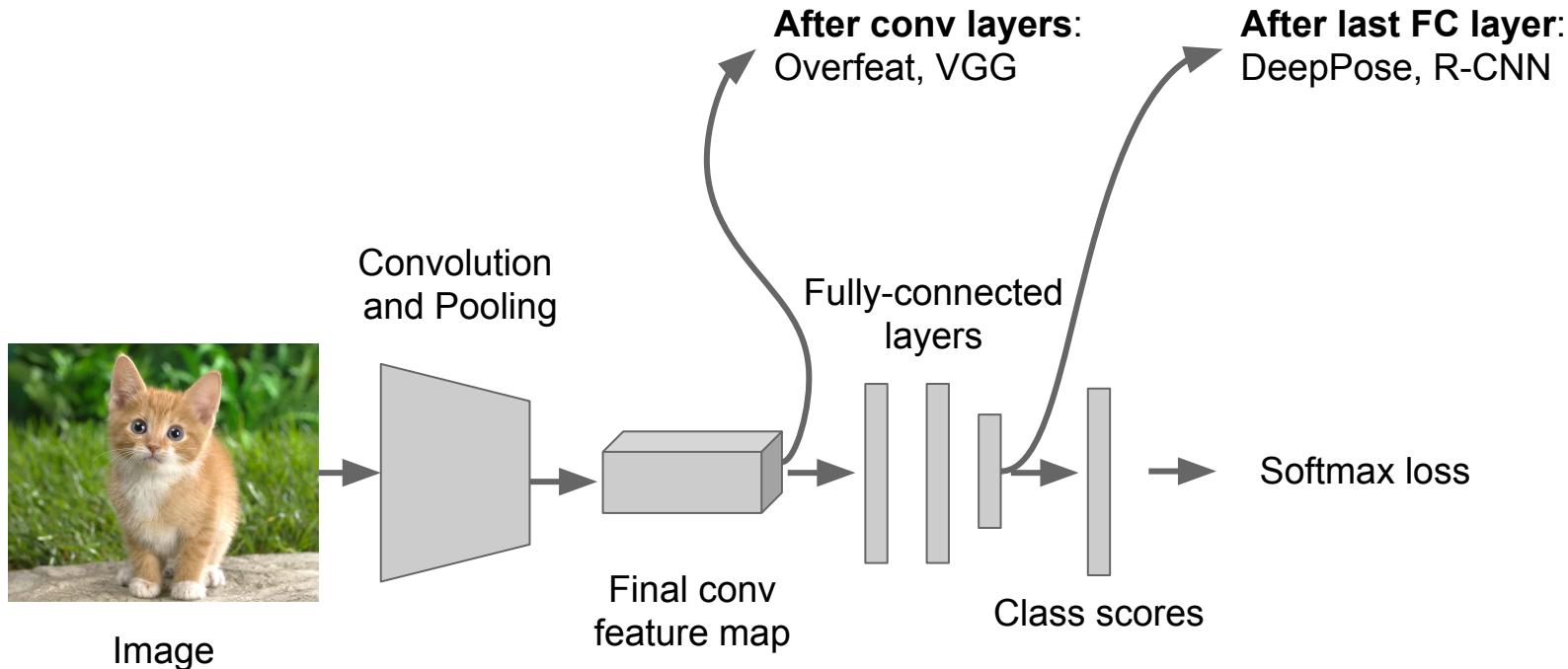


Per-class vs class agnostic regression

Assume classification over C classes:



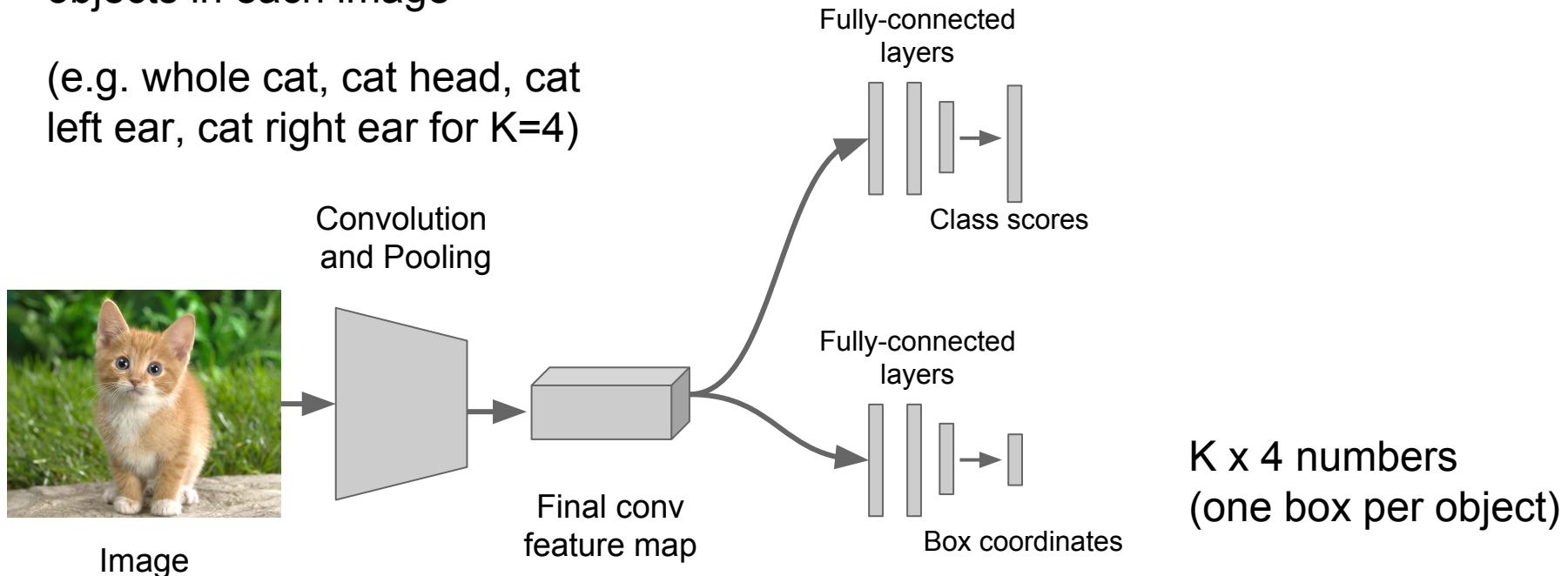
Where to attach the regression head?



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 $K=4$)

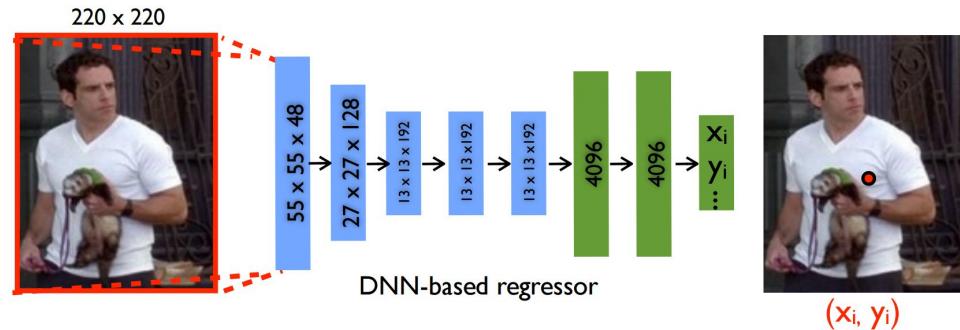


Aside: Human Pose Estimation

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

Very simple

Think if you can use this for projects

Idea #2: Sliding Window

- Run classification + regression network at multiple locations on a high-resolution image
- Convert fully-connected layers into convolutional layers for efficient computation
- Combine classifier and regressor predictions across all scales for final prediction

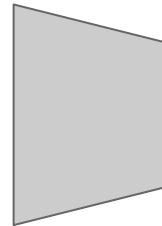
Sliding Window: Overfeat

Winner of ILSVRC 2013
localization challenge

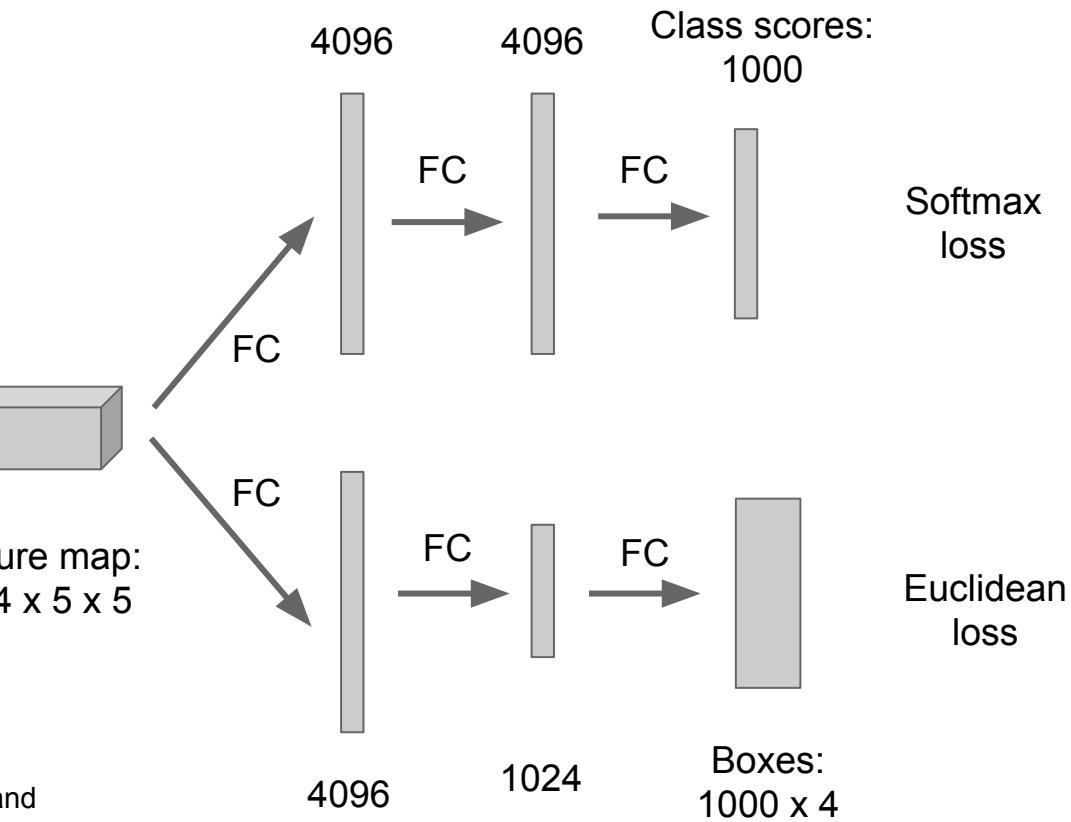


Image:
 $3 \times 221 \times 221$

Convolution
+ pooling

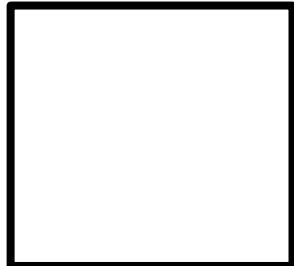


Feature map:
 $1024 \times 5 \times 5$



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

Sliding Window: Overfeat

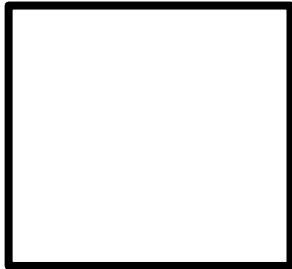


Network input:
 $3 \times 221 \times 221$

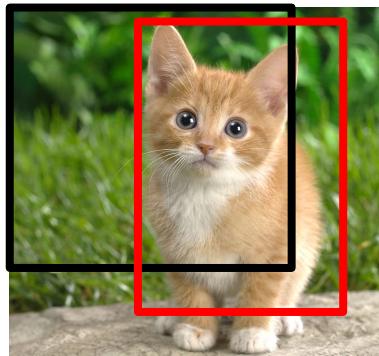


Larger image:
 $3 \times 257 \times 257$

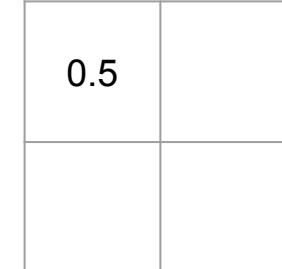
Sliding Window: Overfeat



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

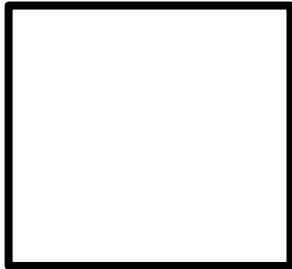


Larger image:
 $3 \times 257 \times 257$

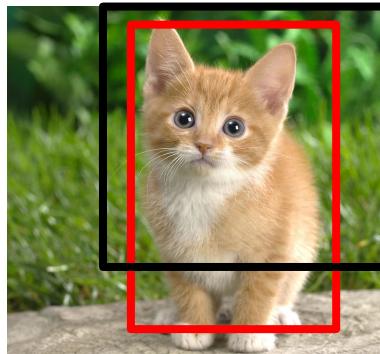


Classification scores:
 $P(\text{cat})$

Sliding Window: Overfeat



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

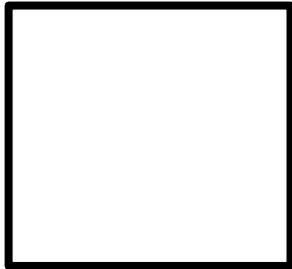


Larger image:
 $3 \times 257 \times 257$

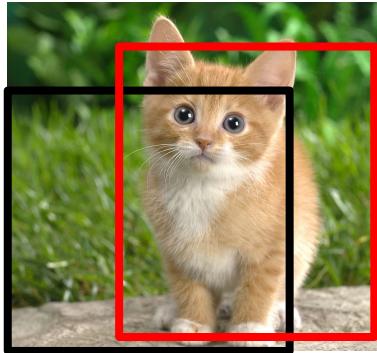
0.5	0.75

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

Sliding Window: Overfeat



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

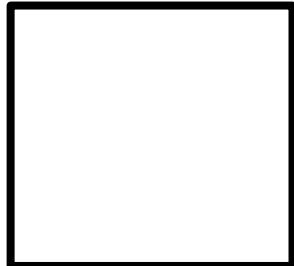


Larger image:
 $3 \times 257 \times 257$

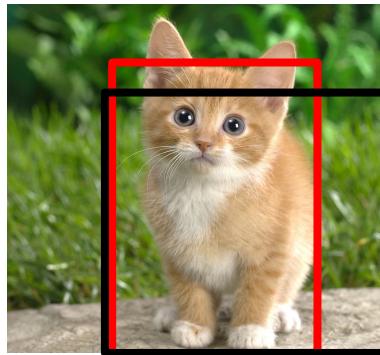
0.5	0.75
0.6	

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

Sliding Window: Overfeat



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

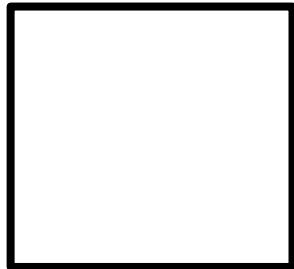


Larger image:
 $3 \times 257 \times 257$

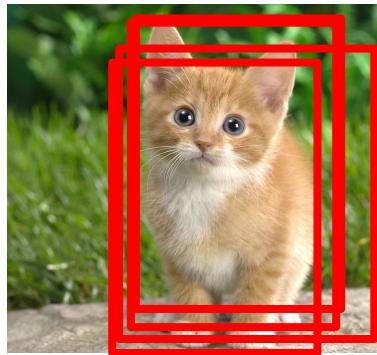
0.5	0.75
0.6	0.8

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

Sliding Window: Overfeat



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

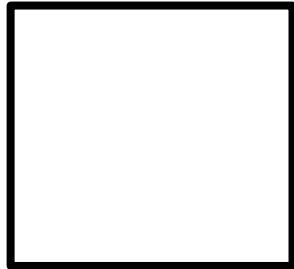


Larger image:
 $3 \times 257 \times 257$

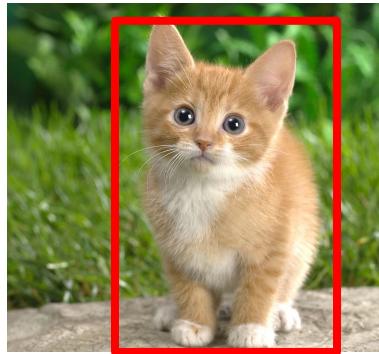
0.5	0.75
0.6	0.8

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

Sliding Window: Overfeat



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



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

Greedily merge boxes and
scores (details in paper)

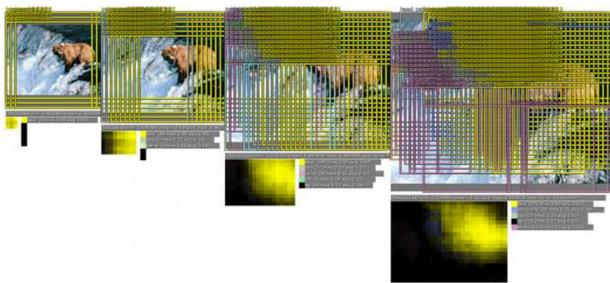
0.8

Classification score: P
(cat)

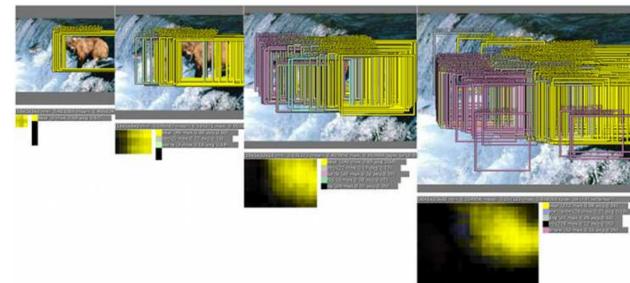
Sliding Window: Overfeat

In practice use many sliding window locations and multiple scales

Window positions + score maps



Box regression outputs



Final Predictions



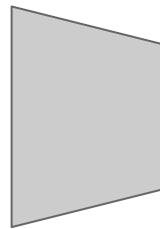
Sermanet et al, "Integrated Recognition, Localization and Detection using Convolutional Networks", ICLR 2014

Efficient Sliding Window: Overfeat

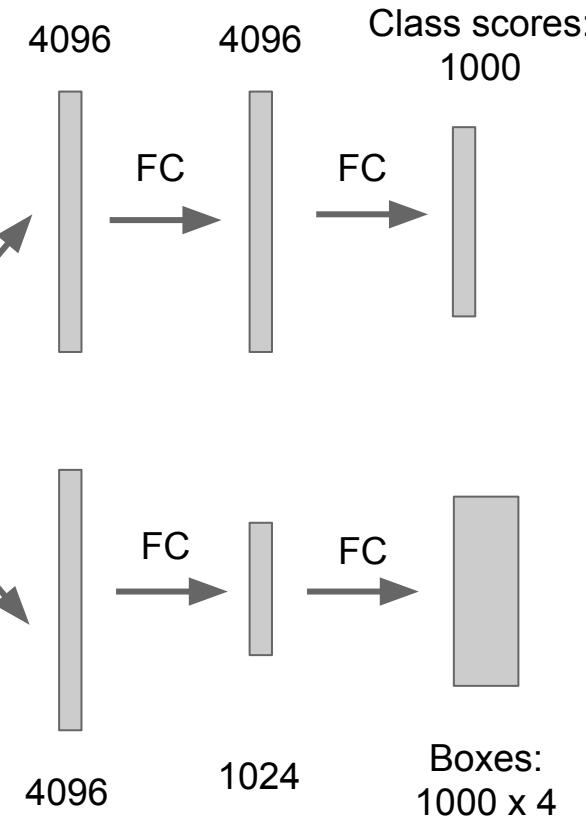


Image:
 $3 \times 221 \times 221$

Convolution
+ pooling



Feature map:
 $1024 \times 5 \times 5$



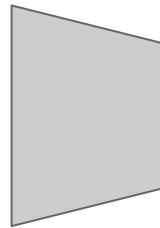
Efficient Sliding Window: Overfeat

Efficient sliding window by converting fully-connected layers into convolutions

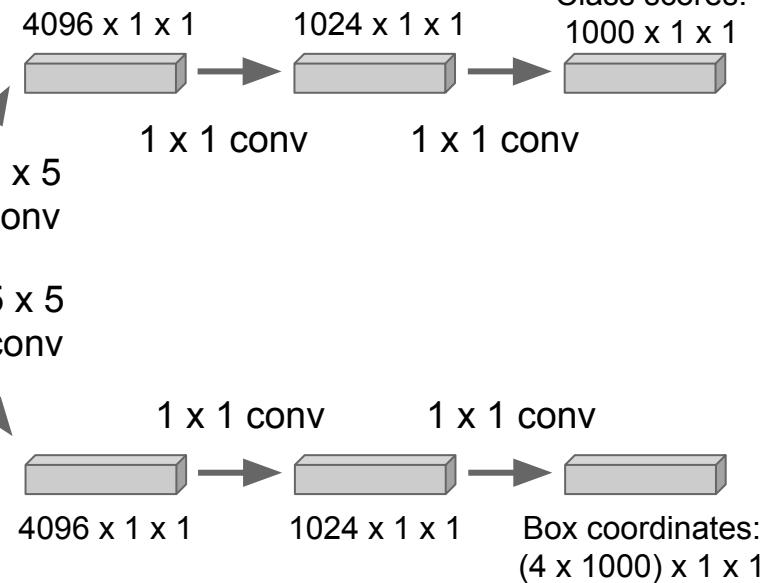


Image:
 $3 \times 221 \times 221$

Convolution
+ pooling

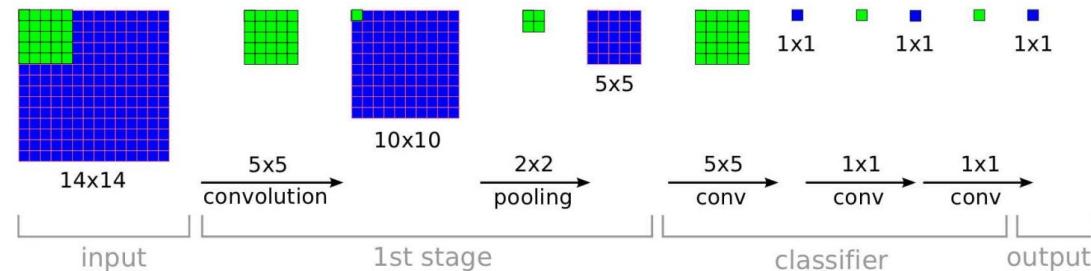


Feature map:
 $1024 \times 5 \times 5$

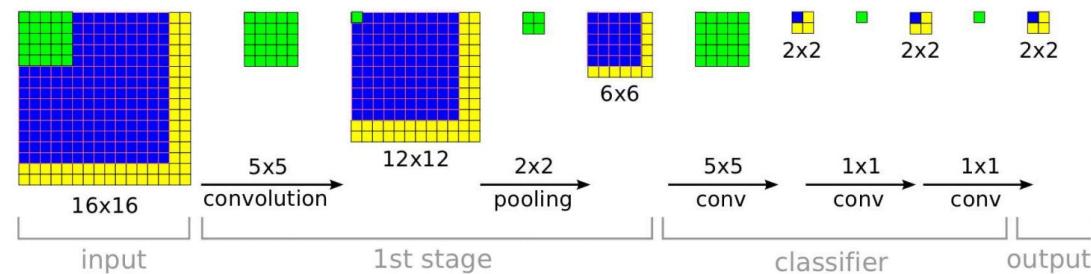


Efficient Sliding Window: Overfeat

Training time: Small image, 1 x 1 classifier output

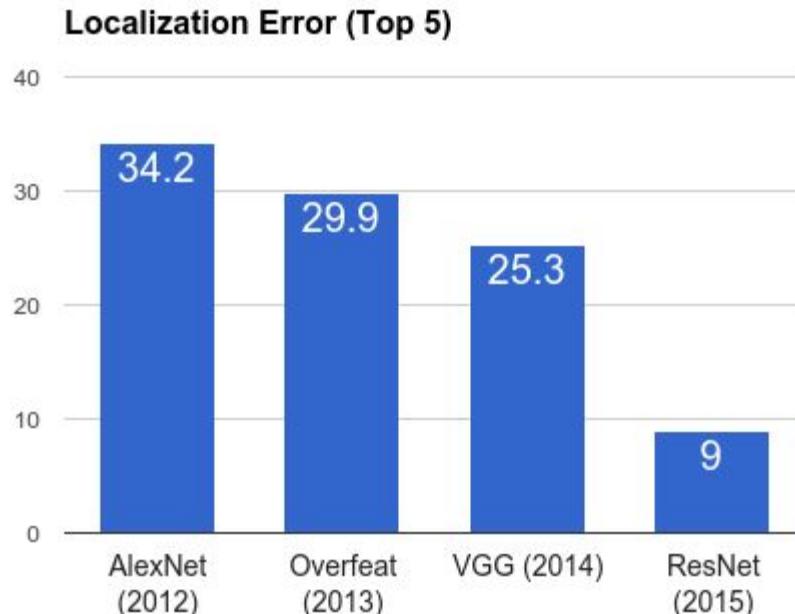


Test time: Larger image, 2 x 2 classifier output, only extra compute at yellow regions



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

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

Computer Vision Tasks

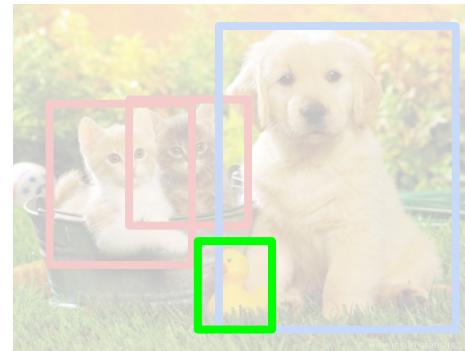
Classification



**Classification
+ Localization**



Object Detection



Instance
Segmentation



Computer Vision Tasks

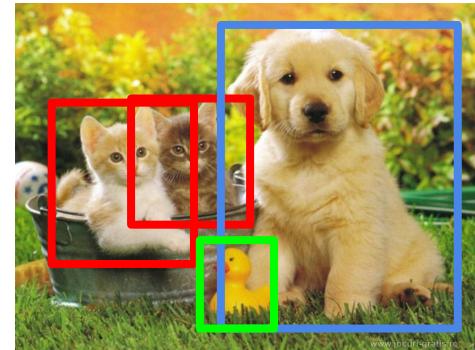
Classification



Classification
+ Localization



Object Detection



Instance
Segmentation



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

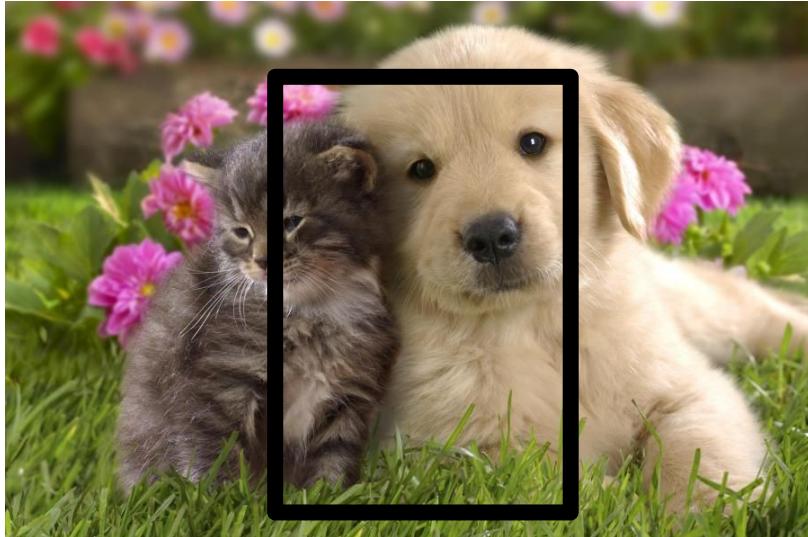
Detection as Classification



CAT? YES!

DOG? NO

Detection as Classification



CAT? NO

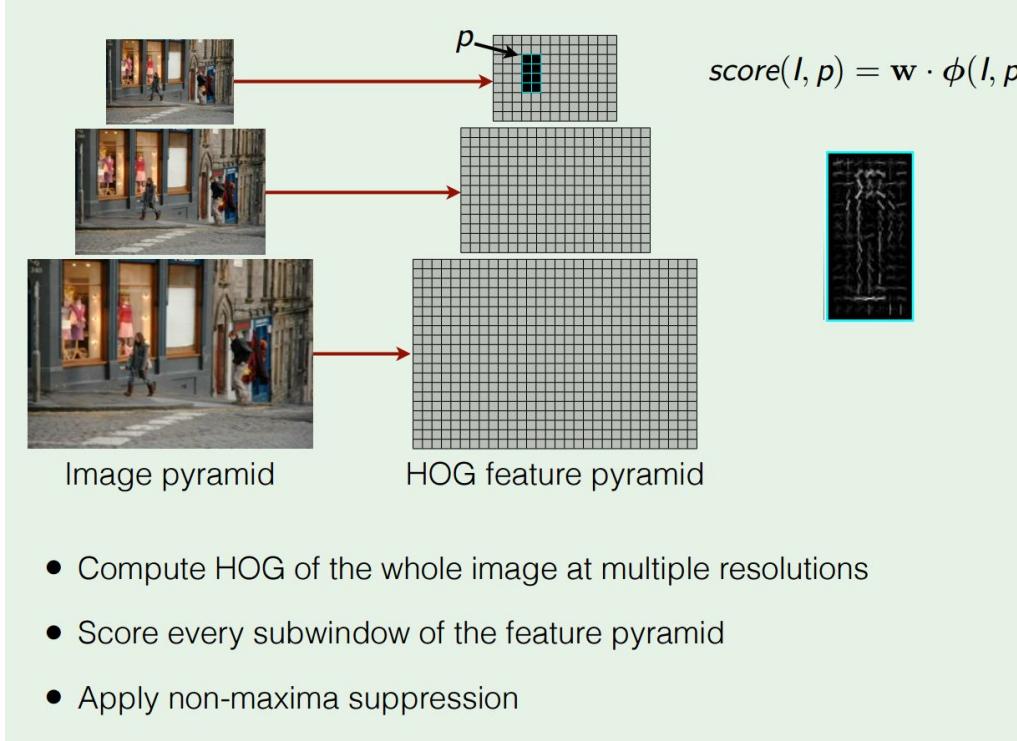
DOG? NO

Detection as Classification

Problem: Need to test many positions and scales

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

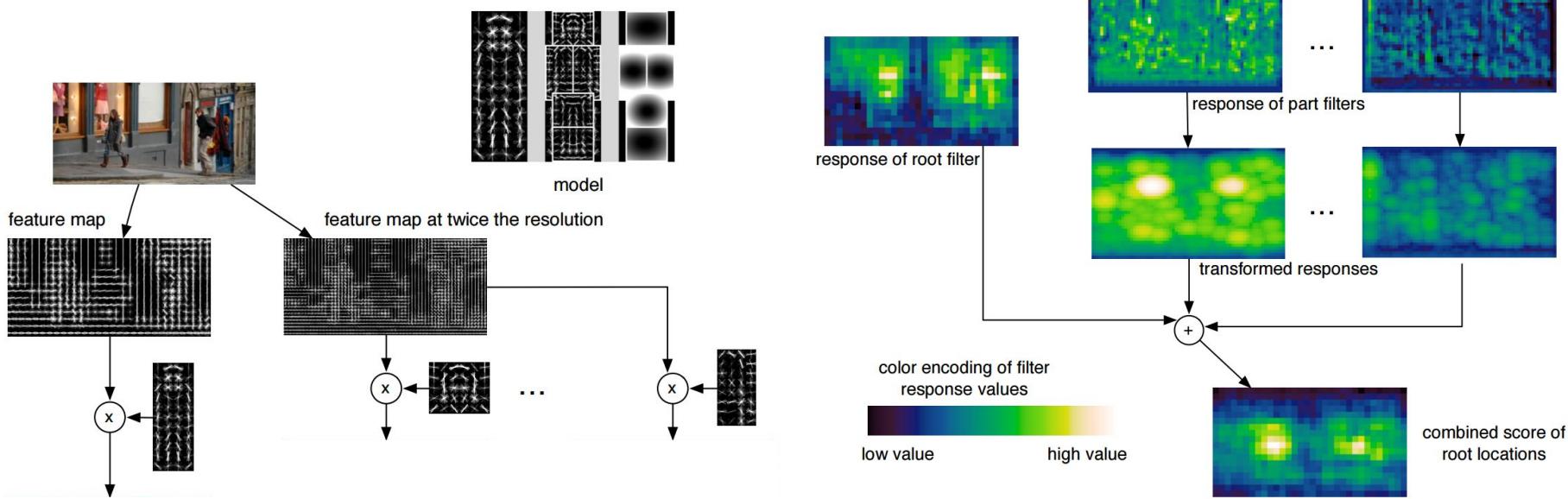
Histogram of Oriented Gradients



Dalal and Triggs, "Histograms of Oriented Gradients for Human Detection", CVPR 2005

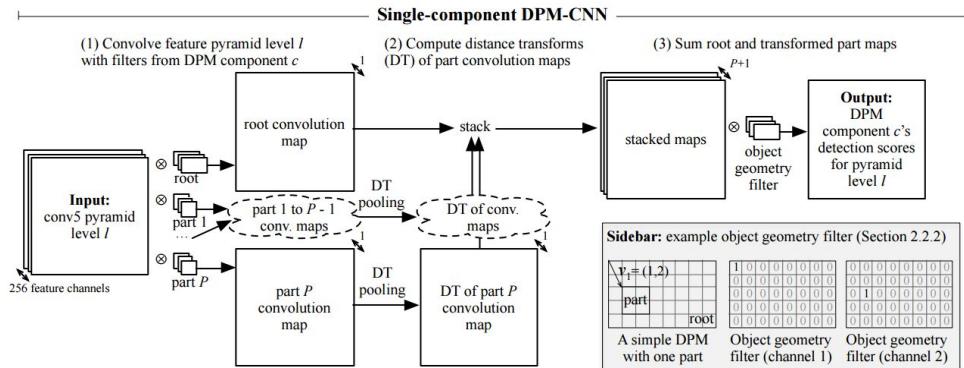
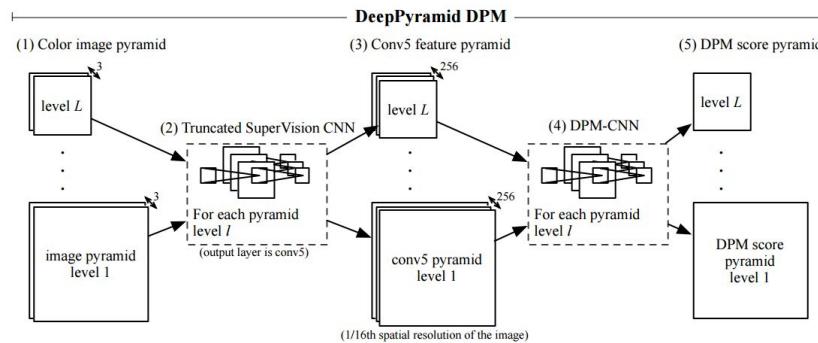
Slide credit: Ross Girshick

Deformable Parts Model (DPM)



Felzenszwalb et al, "Object Detection with Discriminatively Trained Part Based Models", PAMI 2010

Aside: Deformable Parts Models are CNNs?



Girschick et al, "Deformable Part Models are Convolutional Neural Networks", CVPR 2015

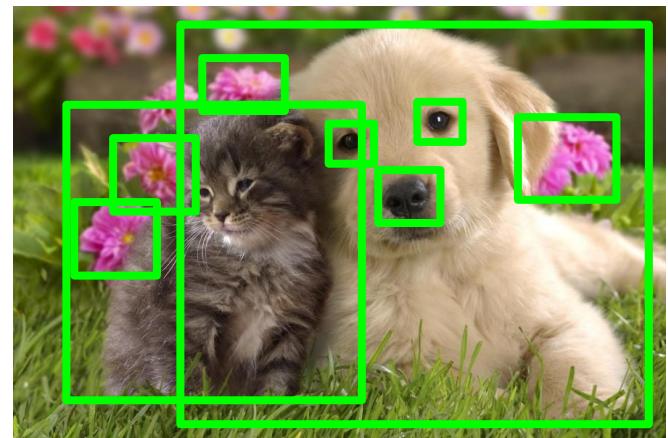
Detection as Classification

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

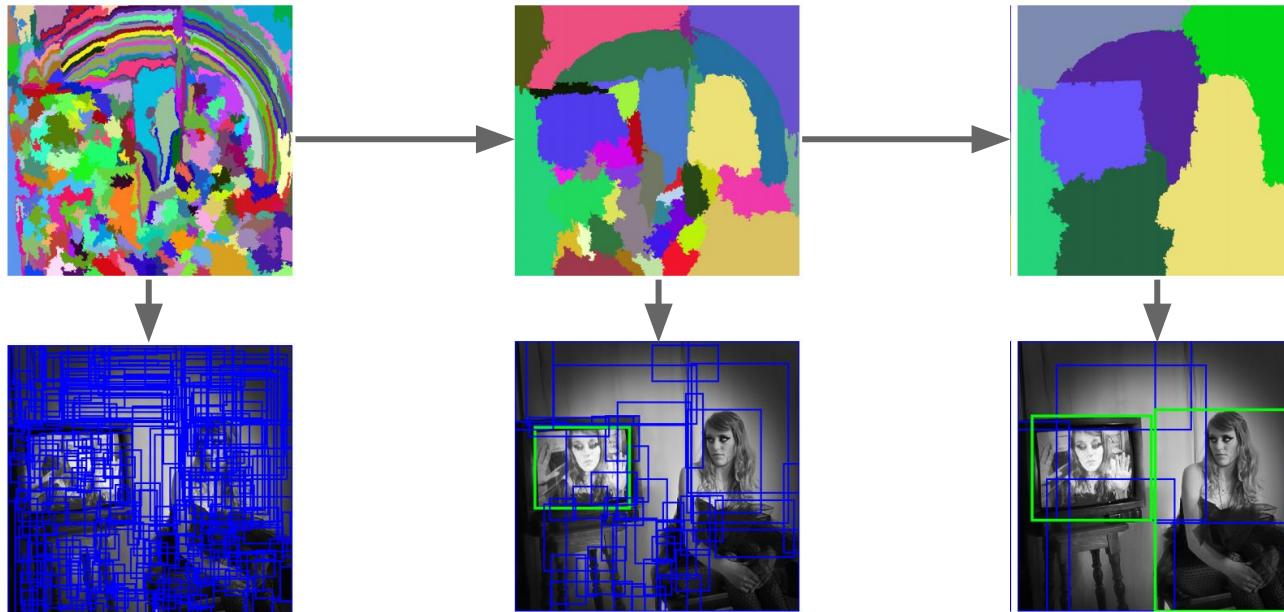
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



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.)	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	★	★	★★
Rantalaankila [27]	Grouping	✓		✓	10	**	·	★★
Rigor [28]	Grouping	✓		✓	10	★	★★	★★
SelectiveSearch [29]	Grouping	✓	✓	✓	10	**	★★★	★★★
Gaussian				✓	0	·	·	★
SlidingWindow				✓	0	***	·	·
Superpixels		✓			1	★	·	·
Uniform				✓	0	·	·	·

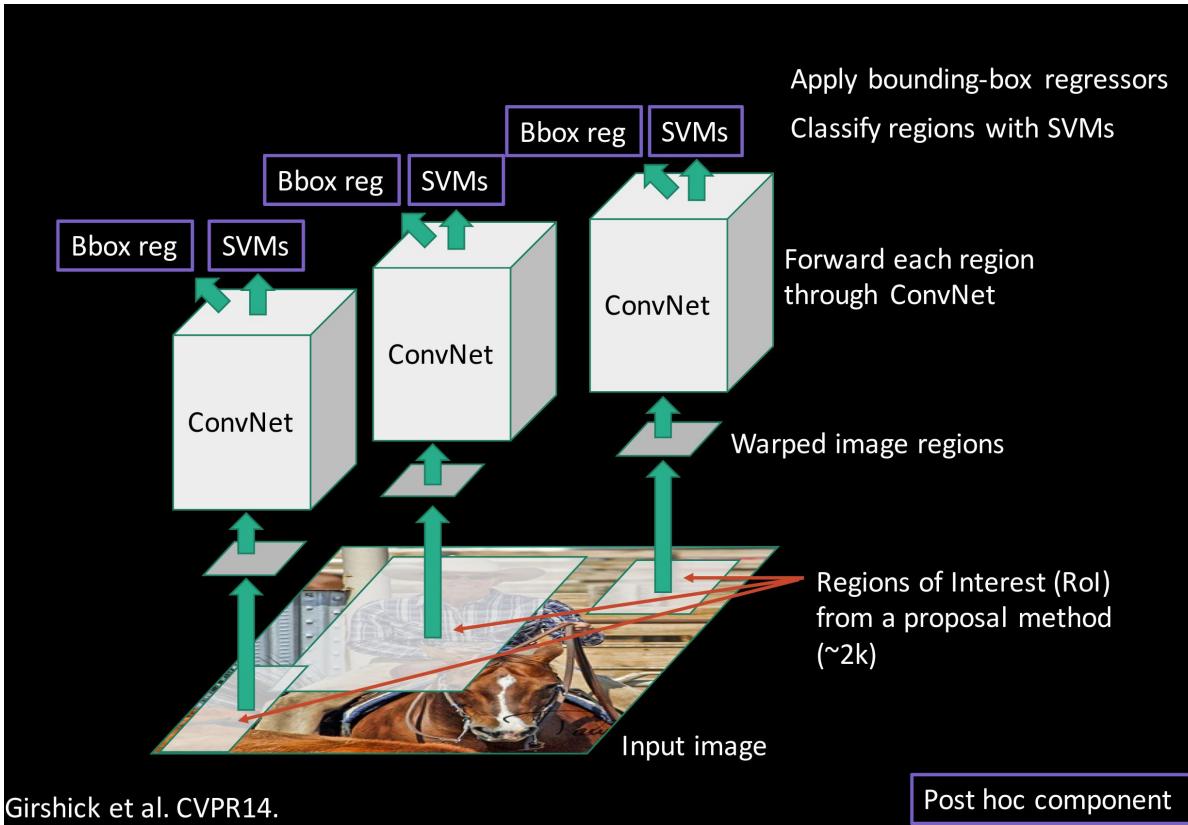
Hosang et al, "What makes for effective detection proposals?", PAMI 2015

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	*	*	**
Rantalaankila [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

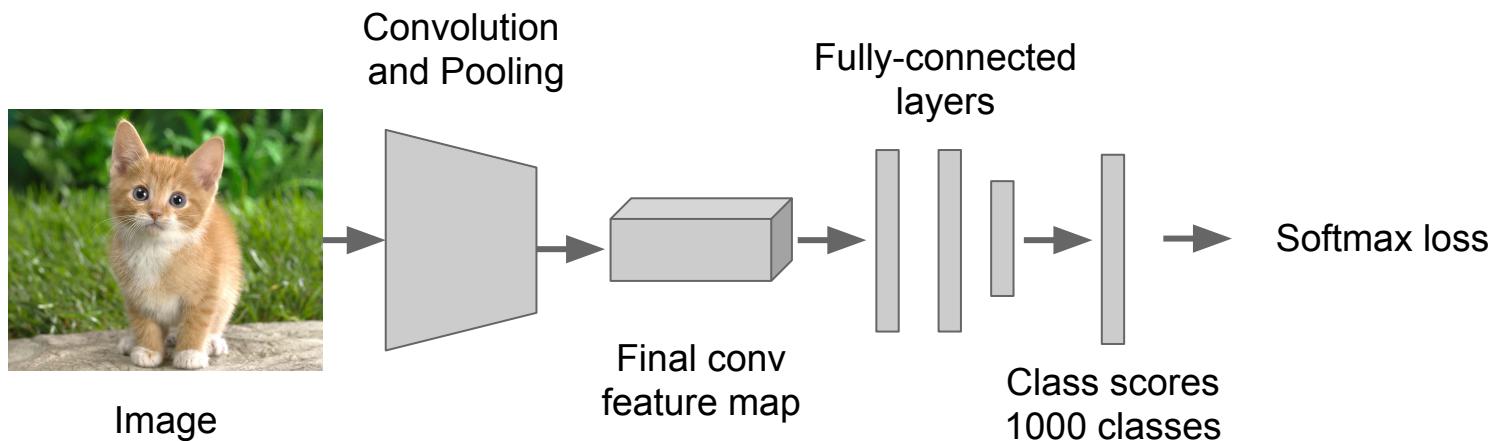


Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation", CVPR 2014

Slide credit: Ross Girshick

R-CNN Training

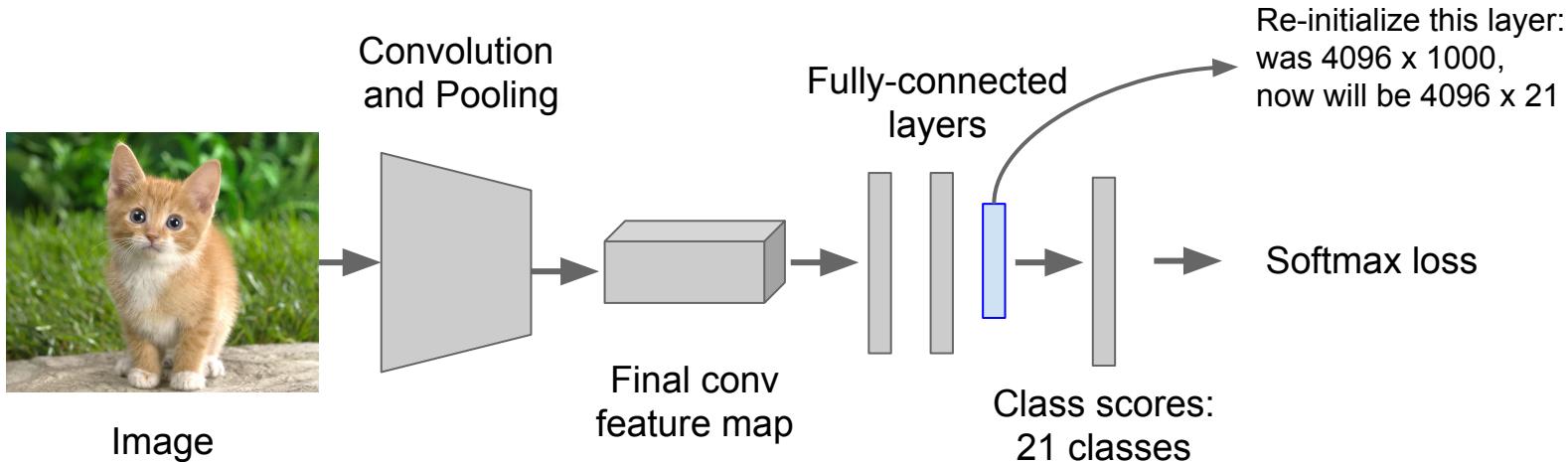
Step 1: Train (or download) a classification model for ImageNet (AlexNet)



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



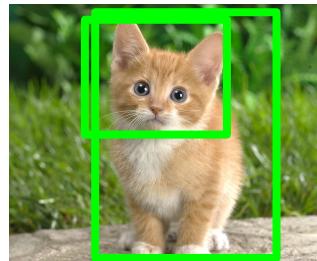
R-CNN Training

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

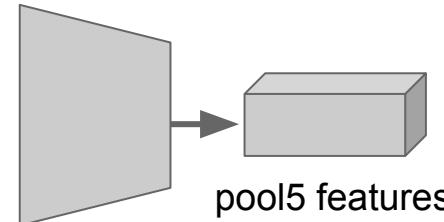


Region Proposals

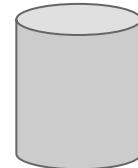


Crop + Warp

Convolution
and Pooling



pool5 features



Save to disk

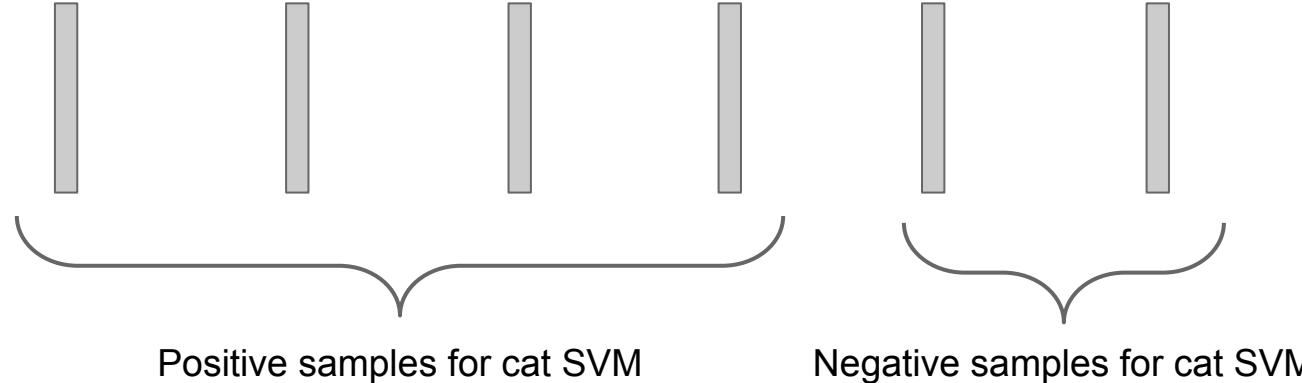
R-CNN Training

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

Training image regions



Cached region features



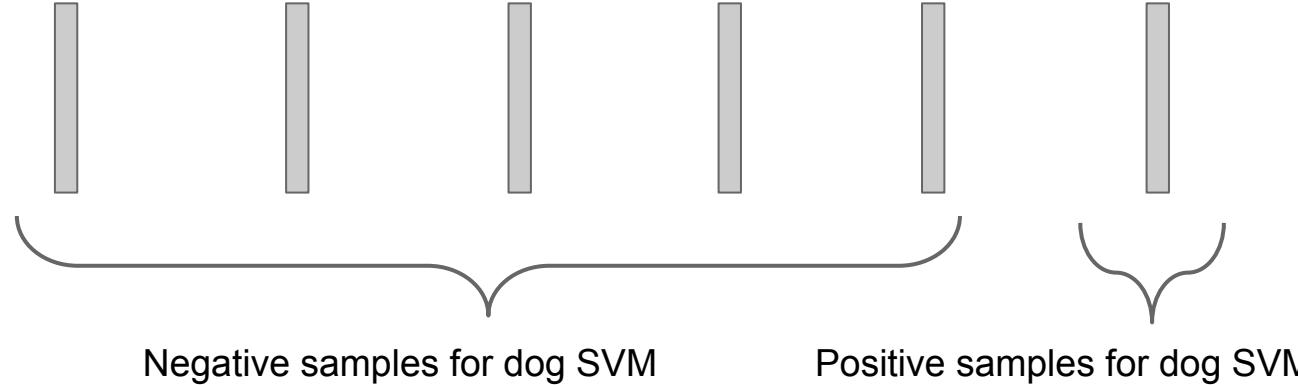
R-CNN Training

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

Training image regions



Cached region features



Negative samples for dog SVM

Positive samples for dog SVM

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	200	80
Number of images (train + val)	~20k	~470k	~120k
Mean objects per image	2.4	1.1	7.2

Object Detection: Evaluation

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

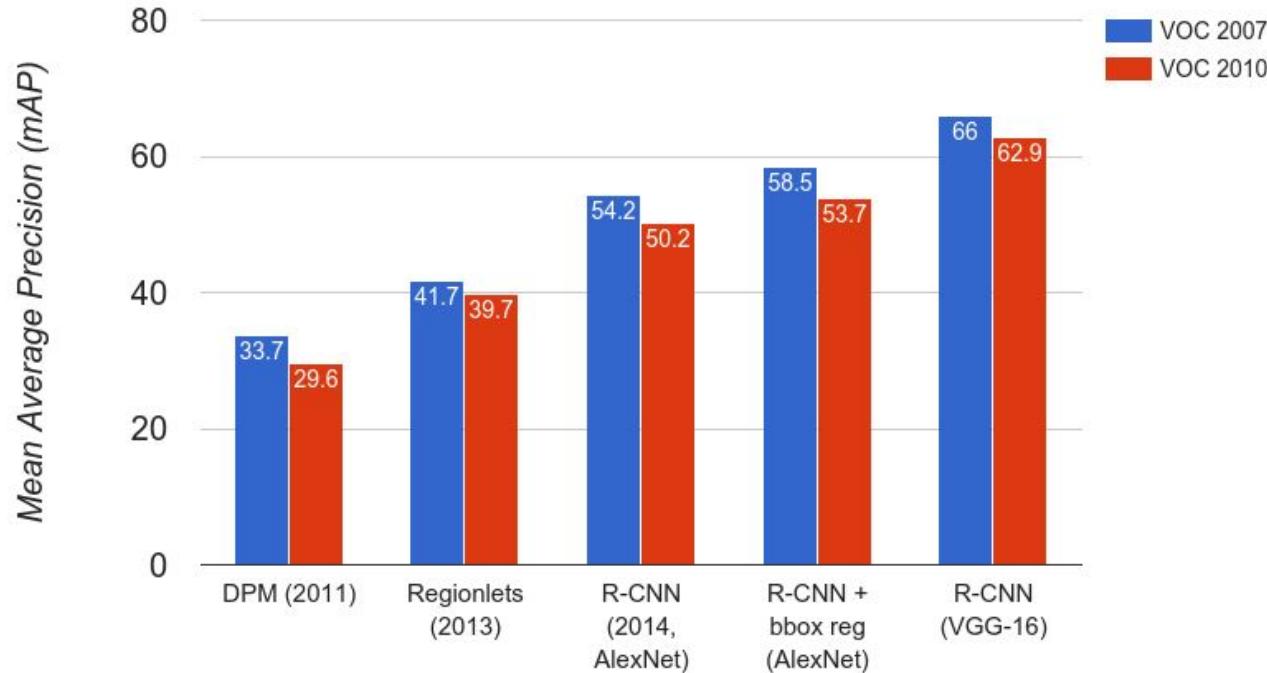
Compute average precision (AP) separately for each class, then average over classes

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

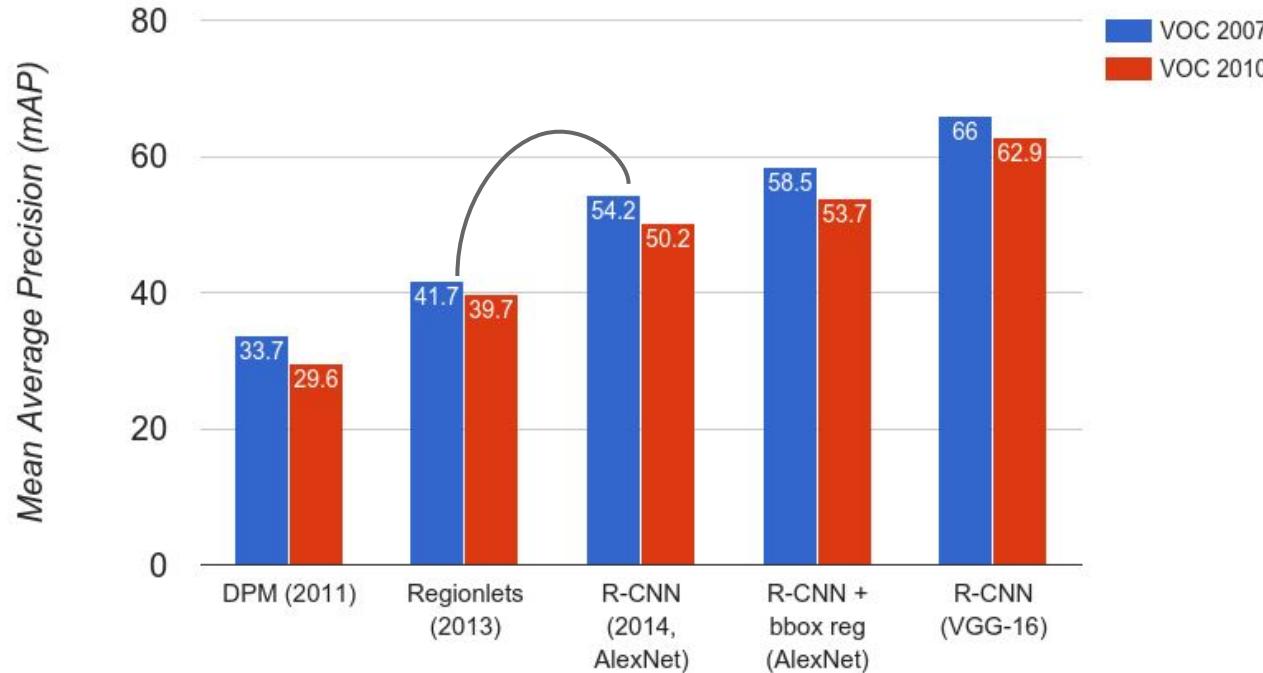
R-CNN Results



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

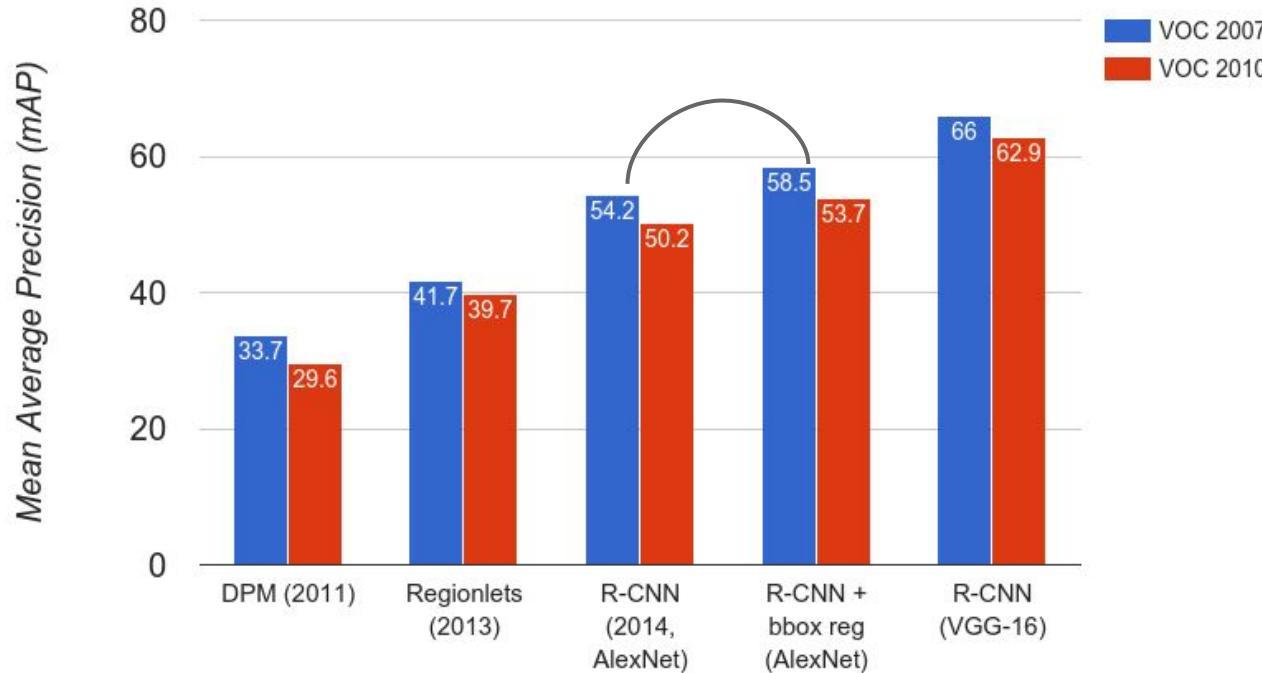
R-CNN Results

Big improvement compared
to pre-CNN methods



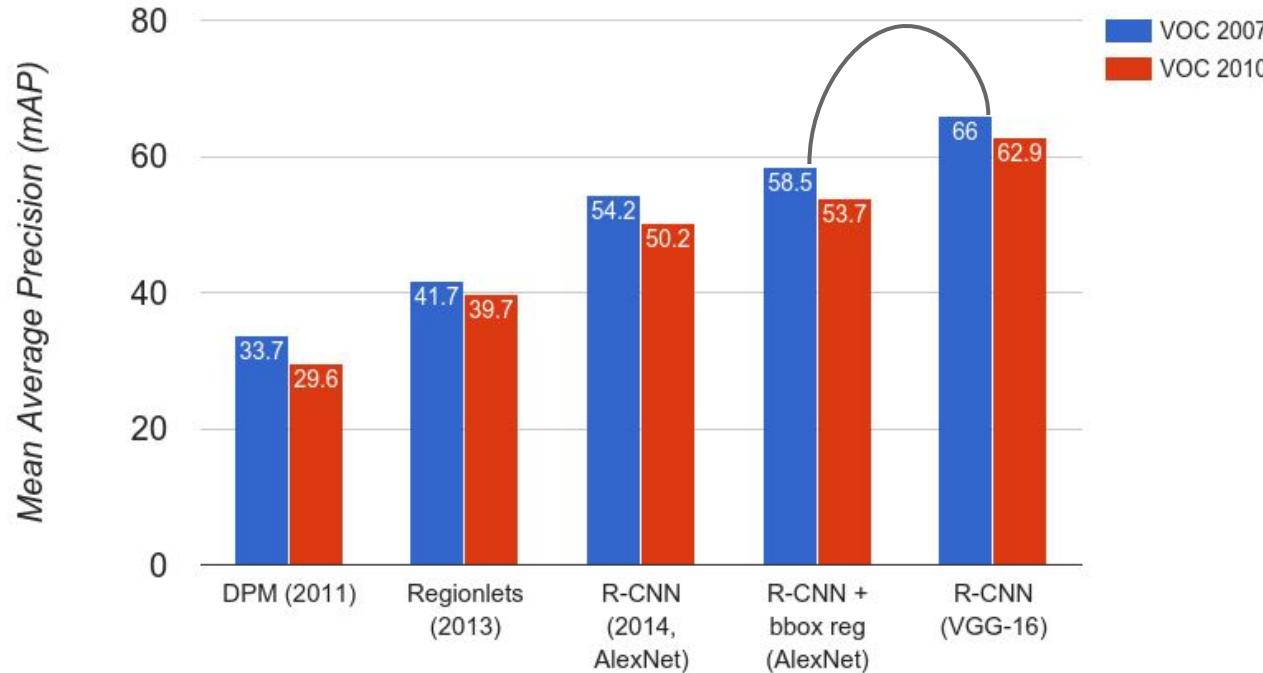
R-CNN Results

Bounding box regression
helps a bit



R-CNN Results

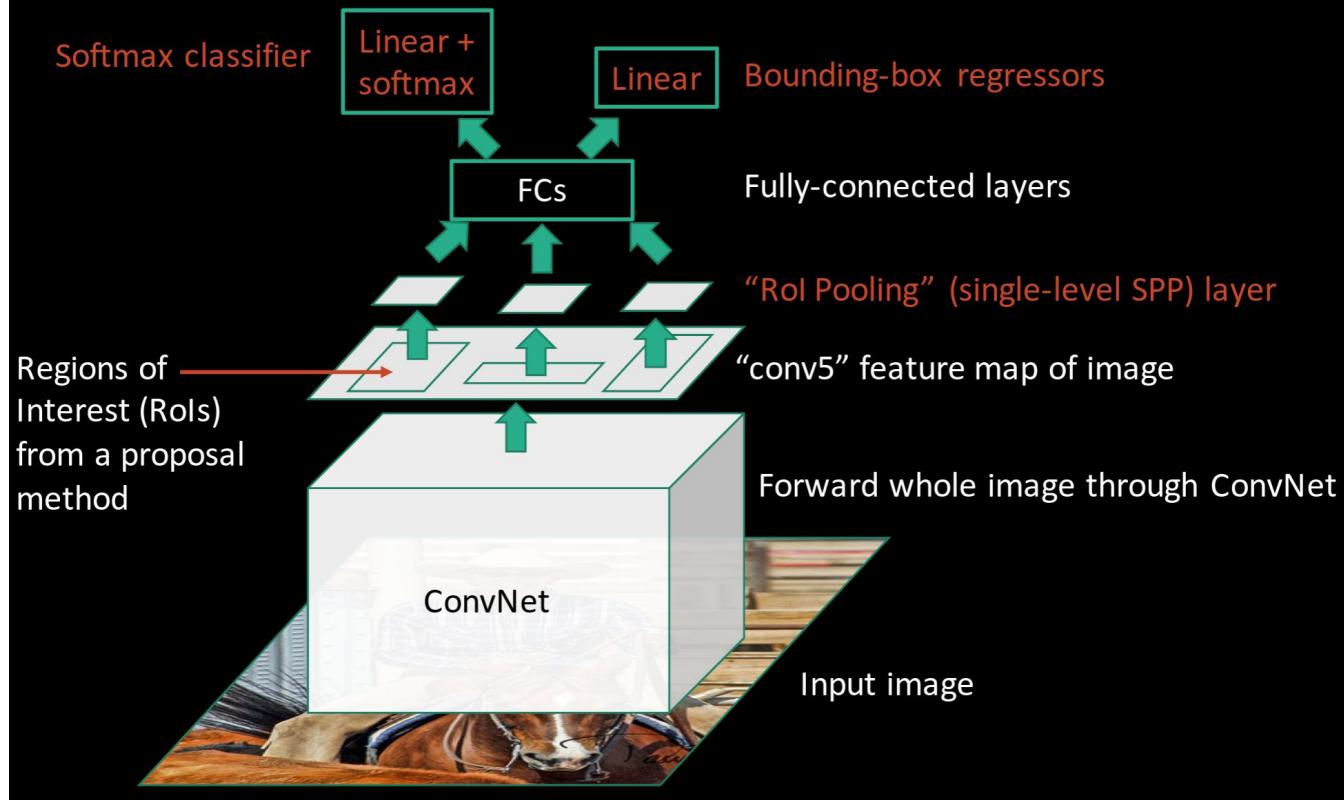
Features from a deeper
network help a lot



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

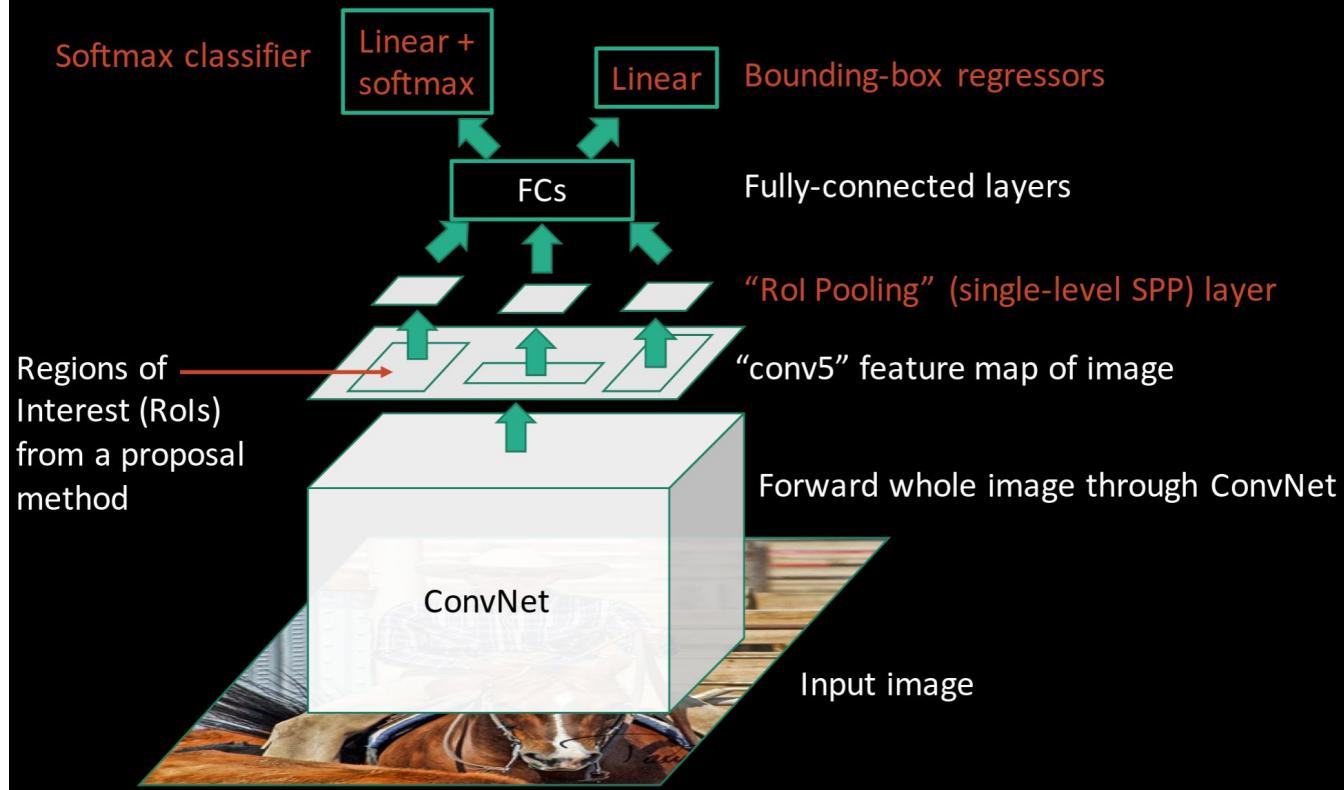
Fast R-CNN (test time)



Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick

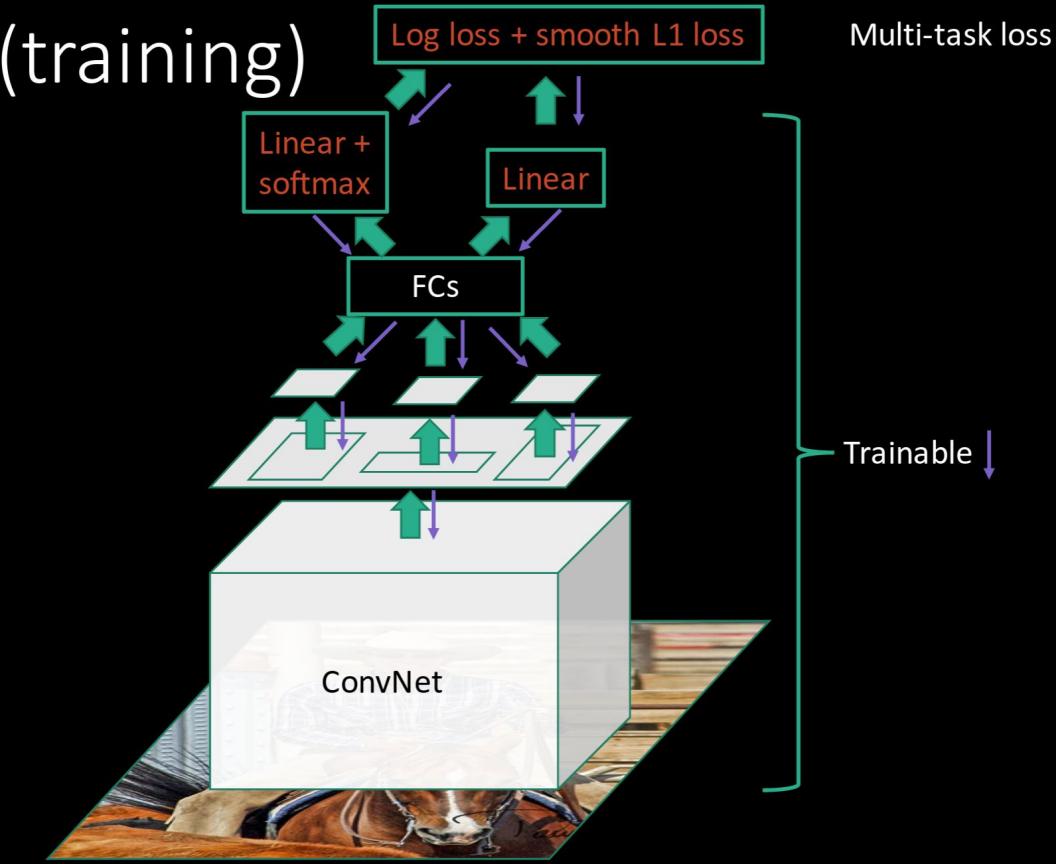
Fast R-CNN (test time)



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!

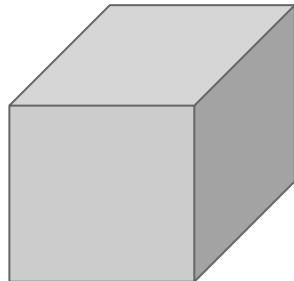
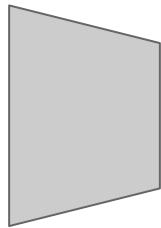
Slide credit: Ross Girshick

Fast R-CNN: Region of Interest Pooling

Convolution
and Pooling

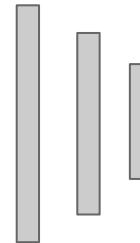


Hi-res input image:
 $3 \times 800 \times 600$
with region
proposal



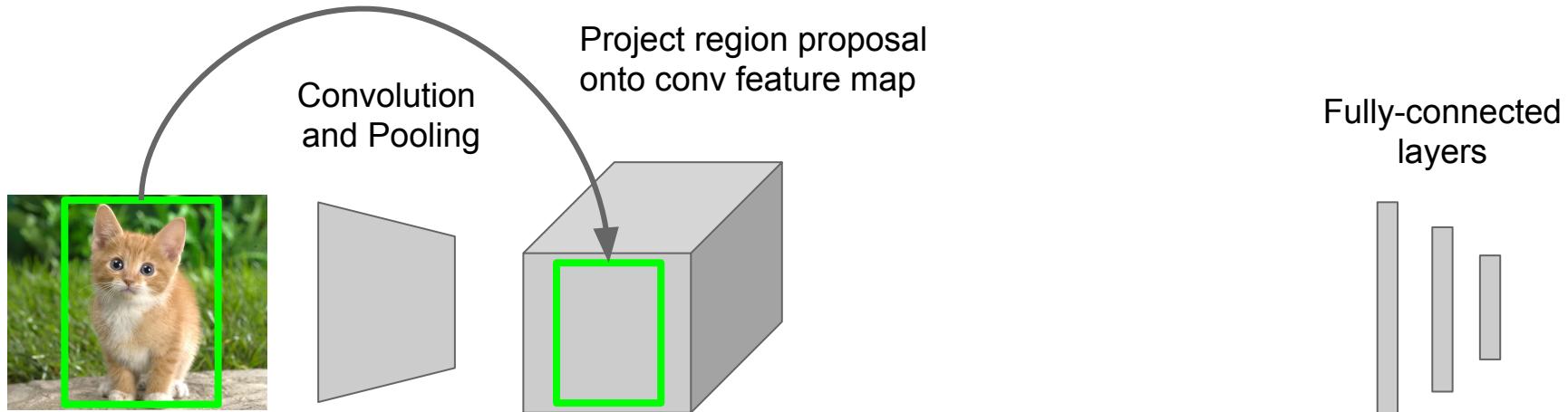
Hi-res conv features:
 $C \times H \times W$
with region proposal

Fully-connected
layers



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

Fast R-CNN: Region of Interest Pooling

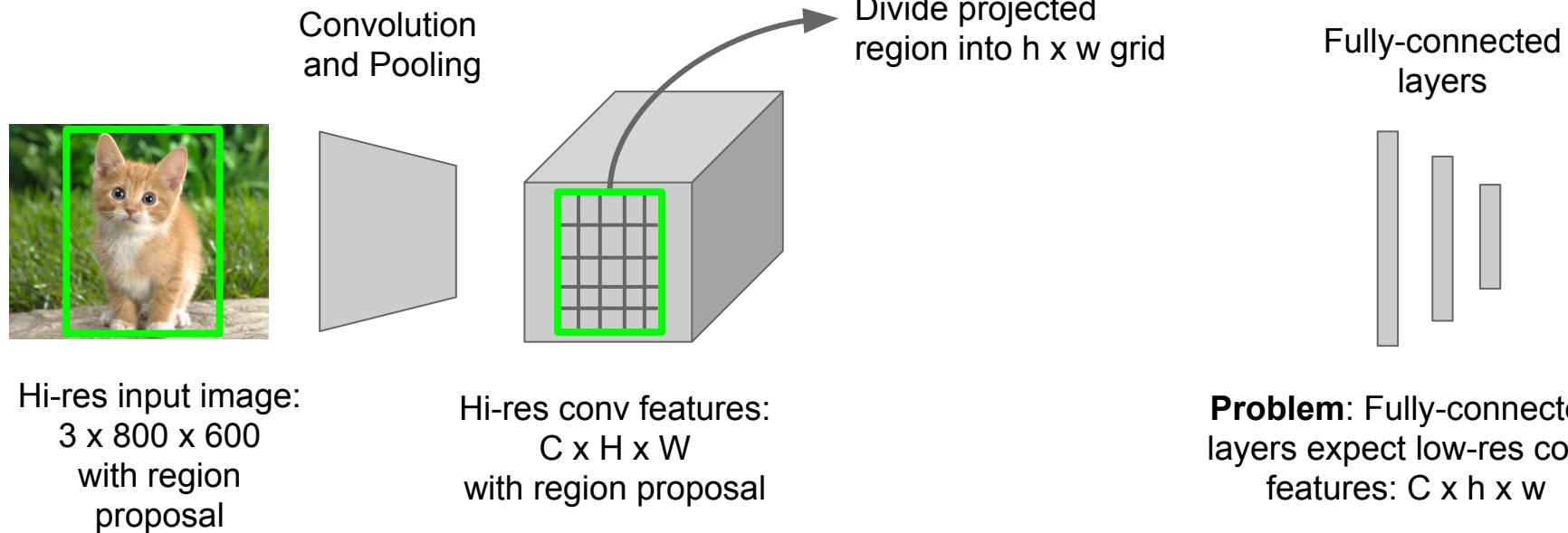


Hi-res input image:
 $3 \times 800 \times 600$
with region
proposal

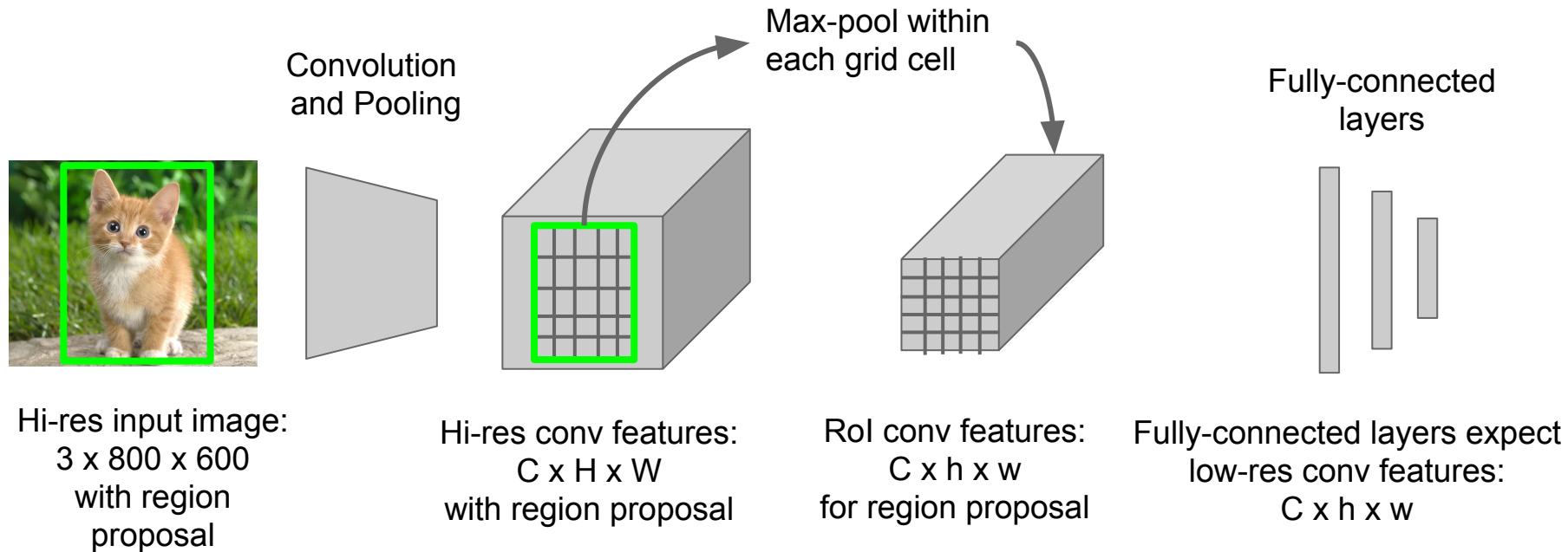
Hi-res conv features:
 $C \times H \times W$
with region proposal

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

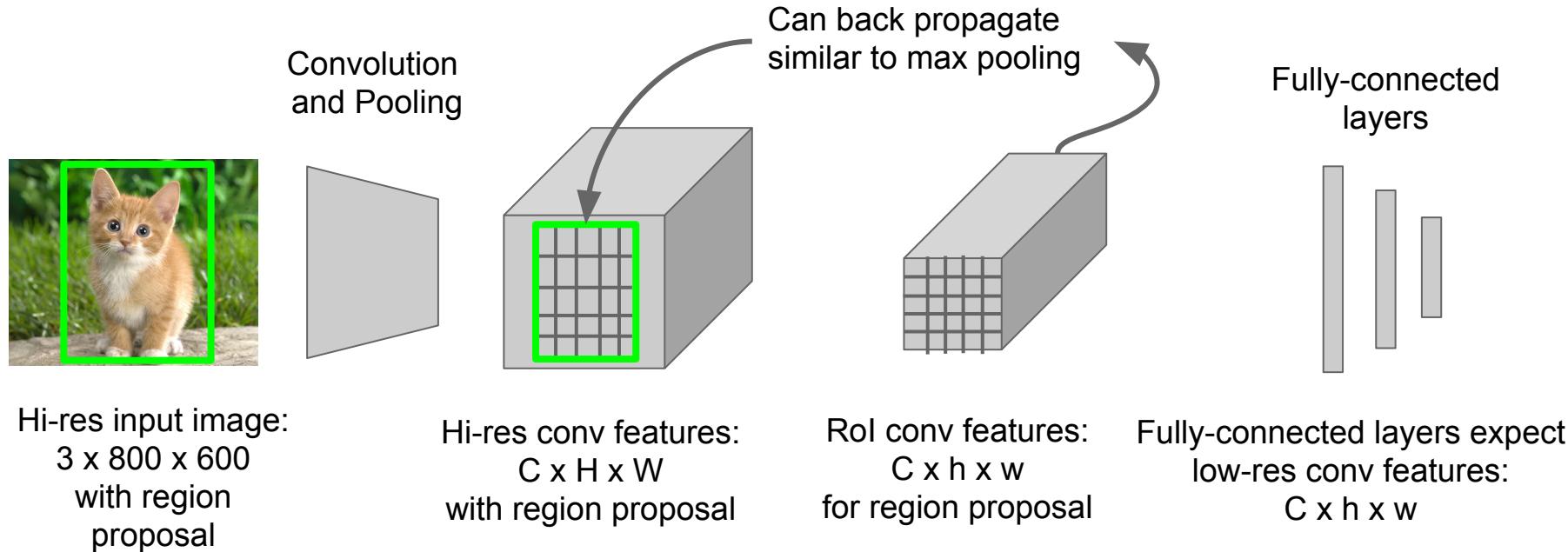
Fast R-CNN: Region of Interest Pooling



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	9.5 hours
(Speedup)	1x	8.8x

Using VGG-16 CNN on Pascal VOC 2007 dataset

Fast R-CNN Results

Faster!

FASTER!

	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

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	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

Fast R-CNN Problem:

Test-time speeds don't include region proposals

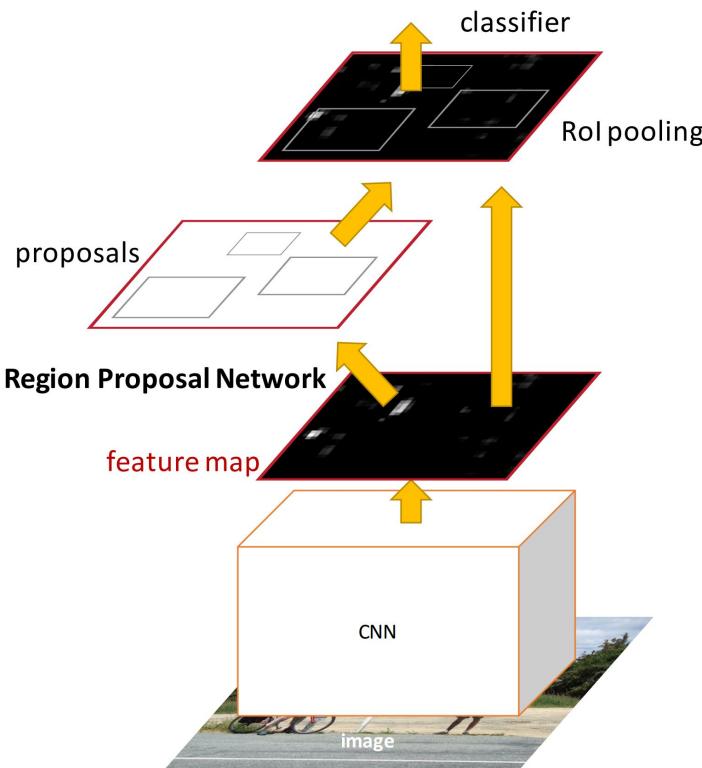
	R-CNN	Fast R-CNN
Test time per image	47 seconds	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

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	0.32 seconds
(Speedup)	1x	146x
Test time per image with Selective Search	50 seconds	2 seconds
(Speedup)	1x	25x

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

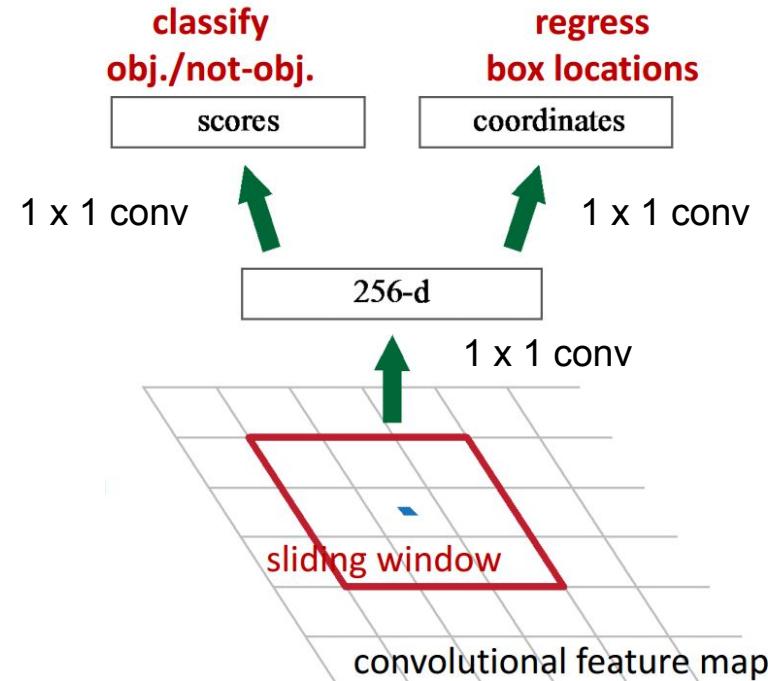
Slide a small window on the feature map

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

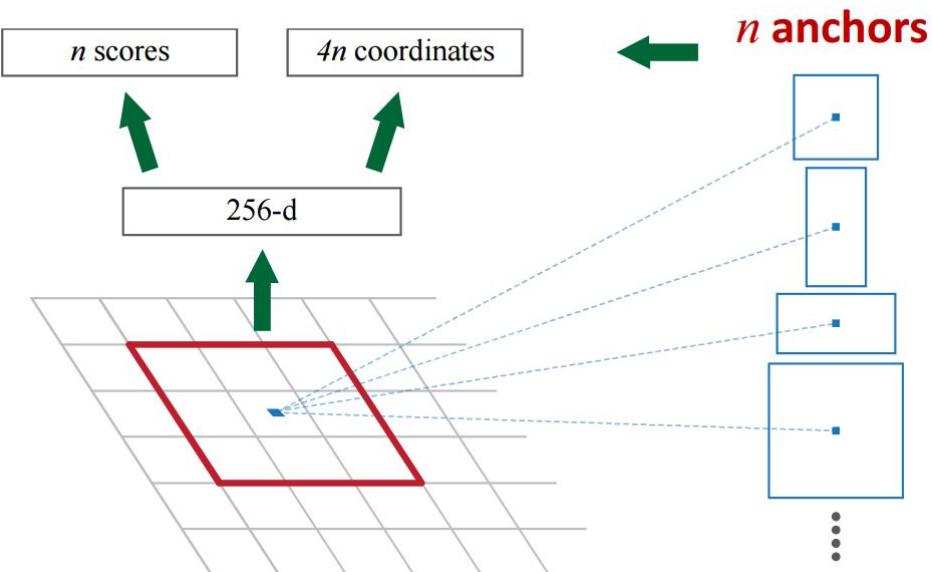
Faster R-CNN: Region Proposal Network

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

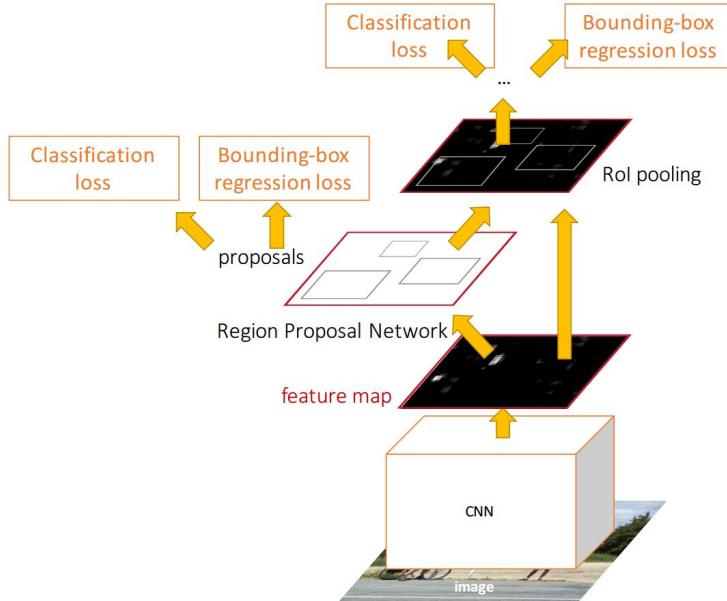
In the paper: Ugly pipeline

- 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

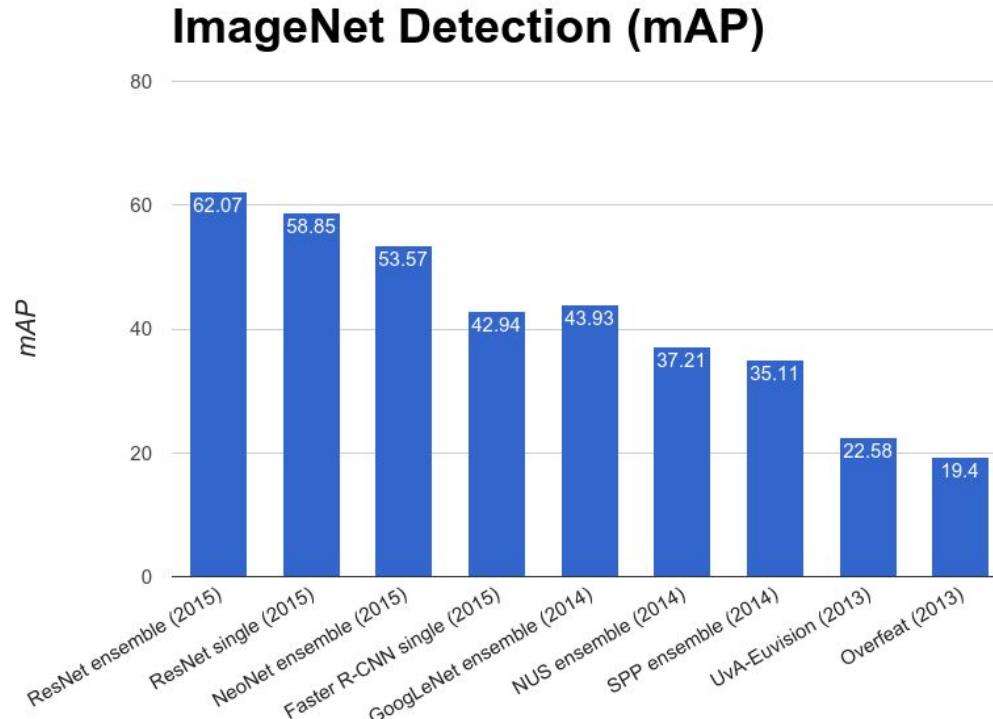
	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image (with proposals)	50 seconds	2 seconds	0.2 seconds
(Speedup)	1x	25x	250x
mAP (VOC 2007)	66.0	66.9	66.9

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	55.7	34.9
ensemble			59.0	37.4

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

ImageNet Detection 2013 - 2015



YOLO: You Only Look Once Detection as Regression

Divide image into $S \times S$ grid

Within each grid cell predict:

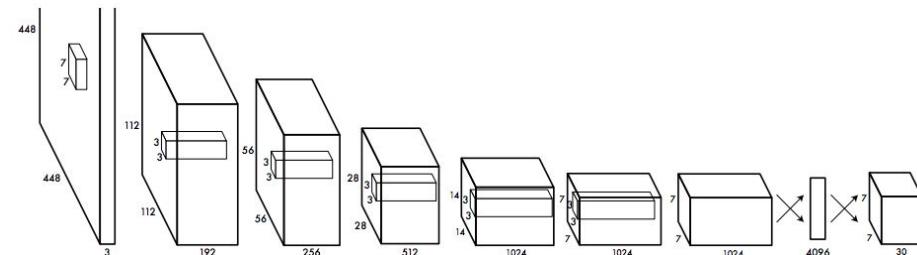
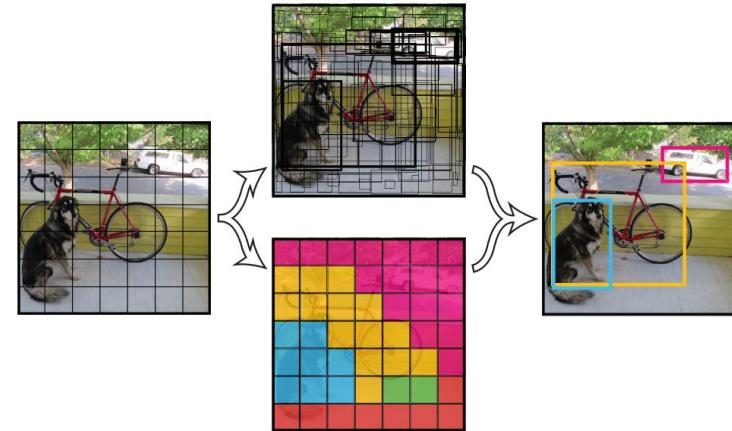
B Boxes: 4 coordinates + confidence

Class scores: C numbers

Regression from image to
 $7 \times 7 \times (5 * B + C)$ 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

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

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

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

(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 \rightarrow 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