Object detection and localization

Computer Vision Tasks

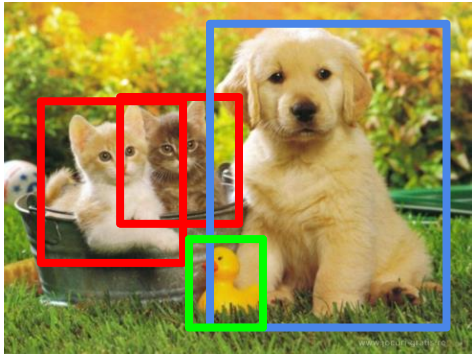
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation

CAT, DOG, DUCK

Single object

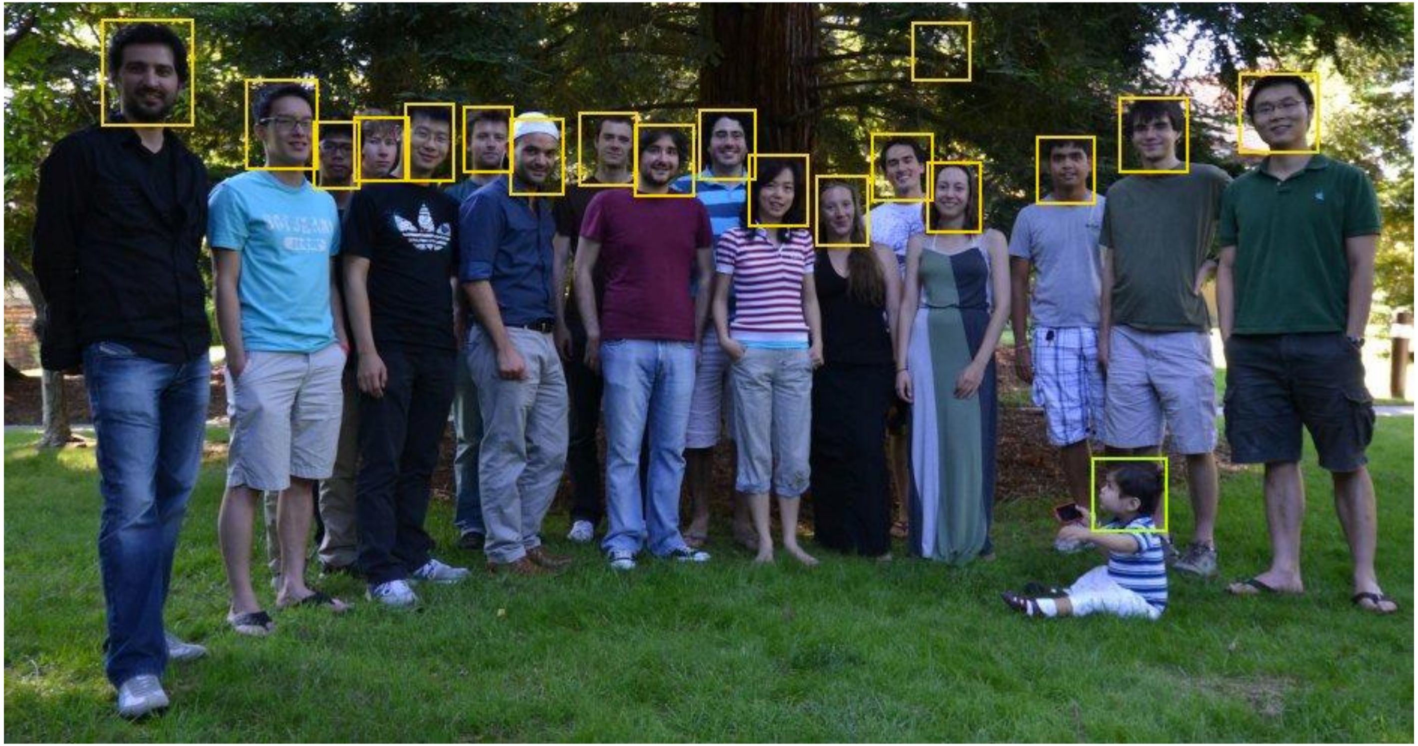
Multiple objects

Faces

- Face detection
- One category: face
- Frontal faces
- Fairly rigid, unoccluded

Human Face Detection in Visual Scenes. H. Rowley, S. Baluja, T. Kanade. 1995.

Faces

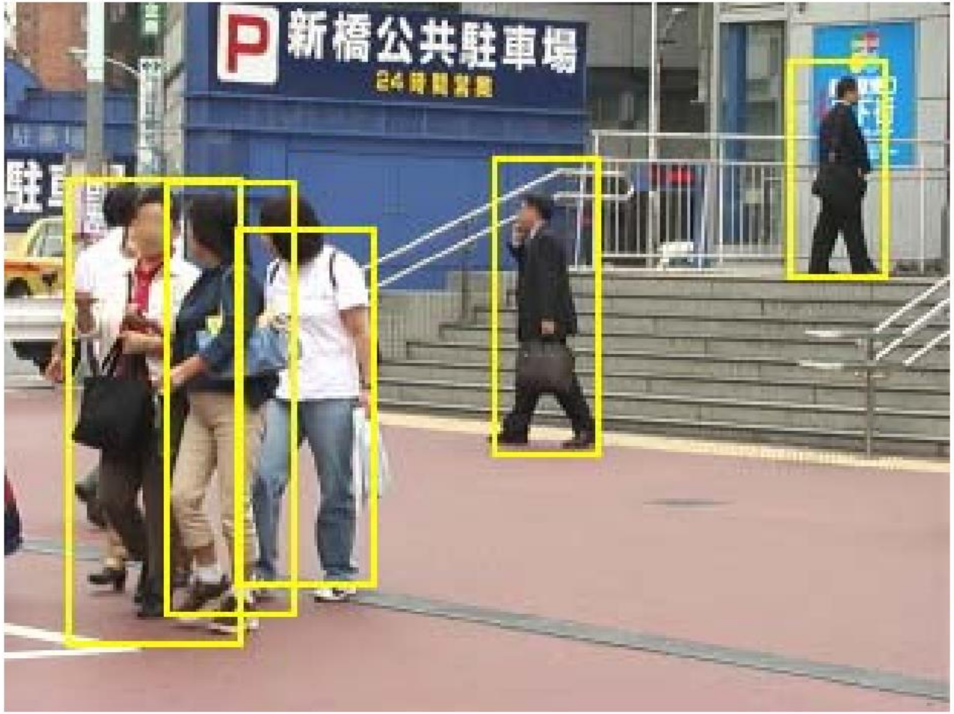


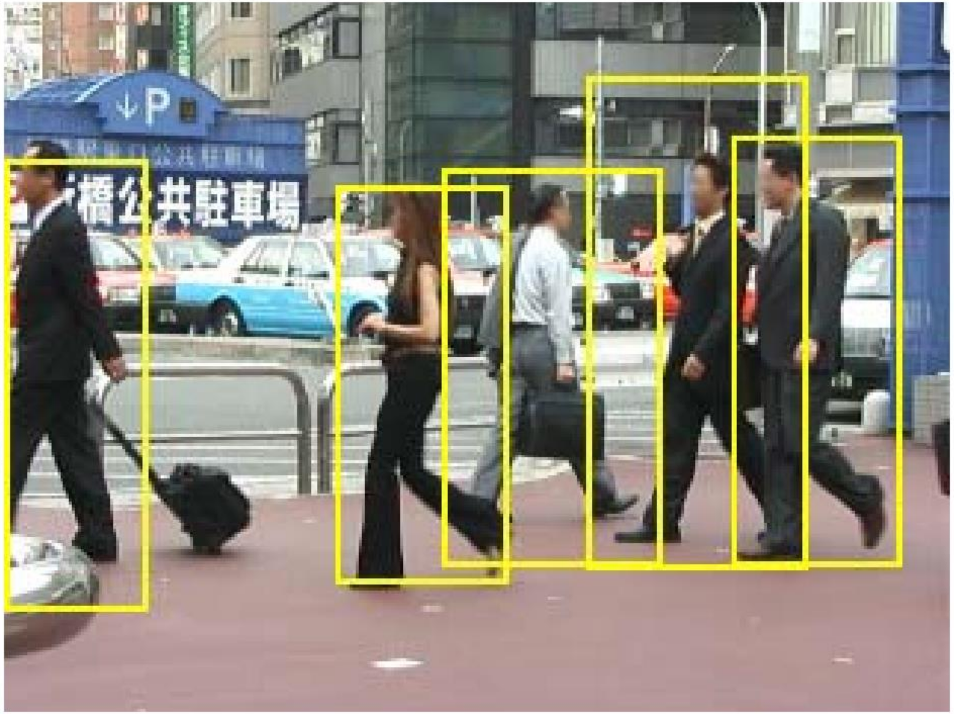
Pedestrians

- One category: pedestrians
- Slight pose variations and small distortions
- Partial occlusions

Histograms of Oriented Gradients for Human Detection. N. Dalal and B. Triggs. CVPR 2005

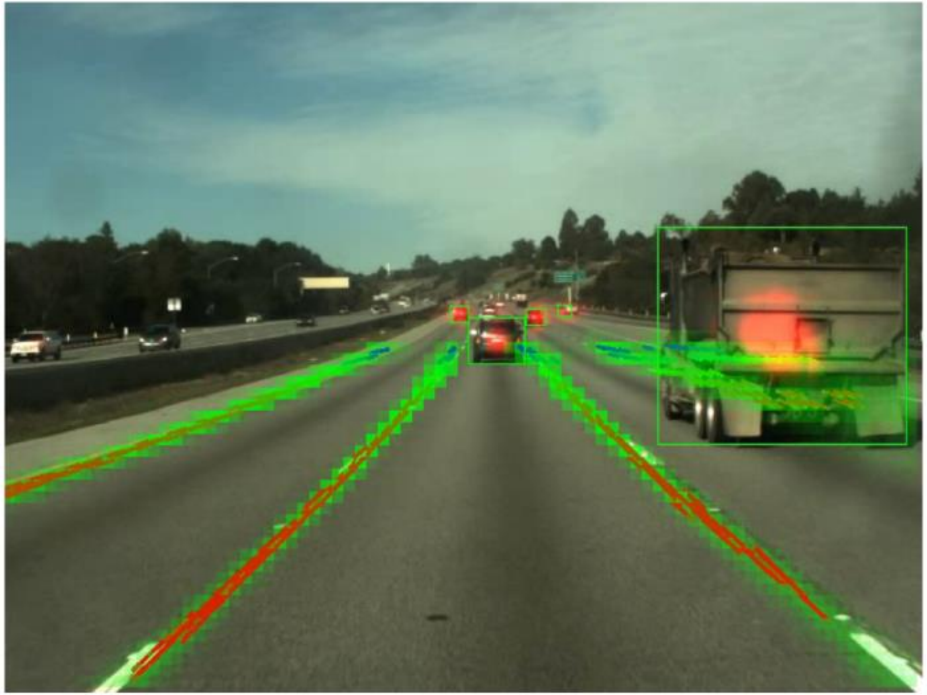
Pedestrians



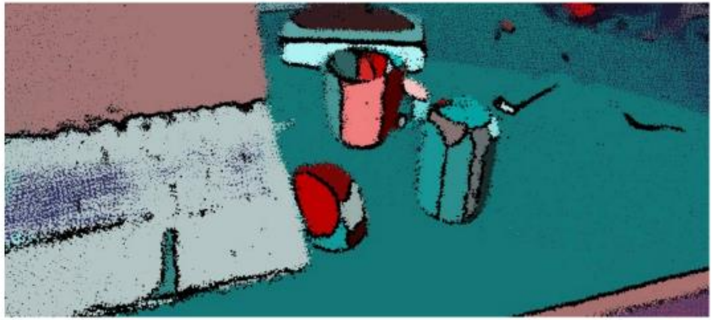


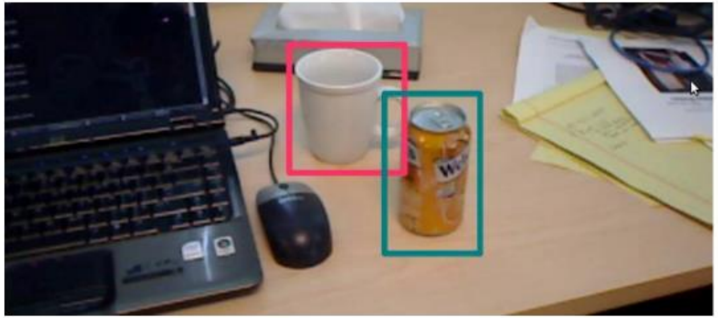
Applications: Tagging People

Applications: Autonomous Driving



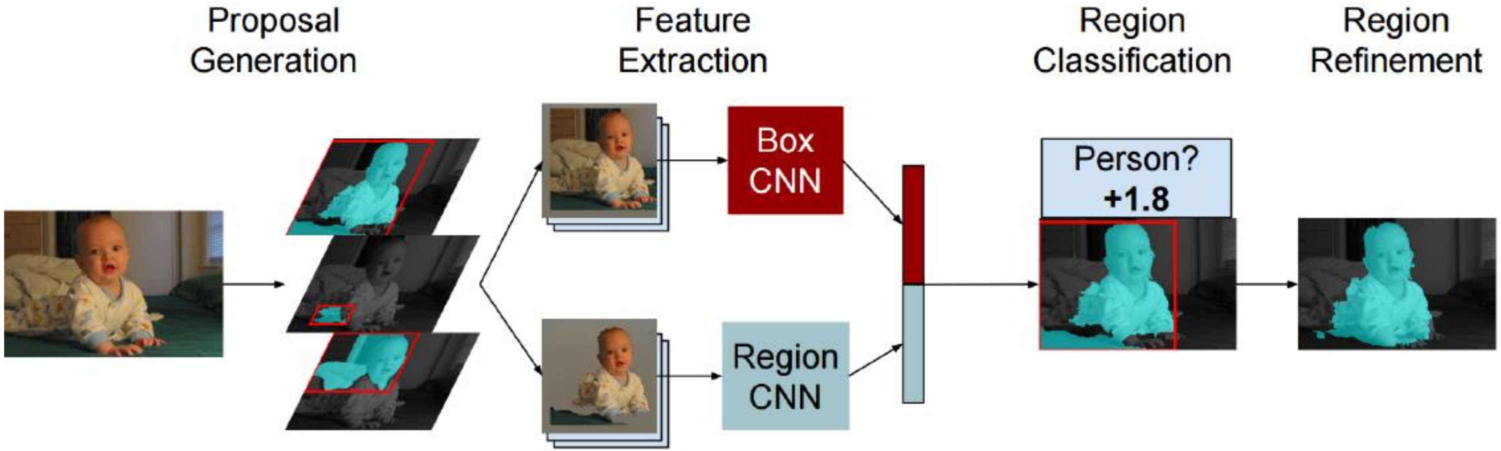
Applications: Robotics





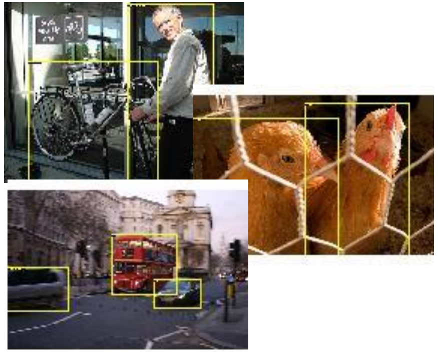
Applications: Tracking

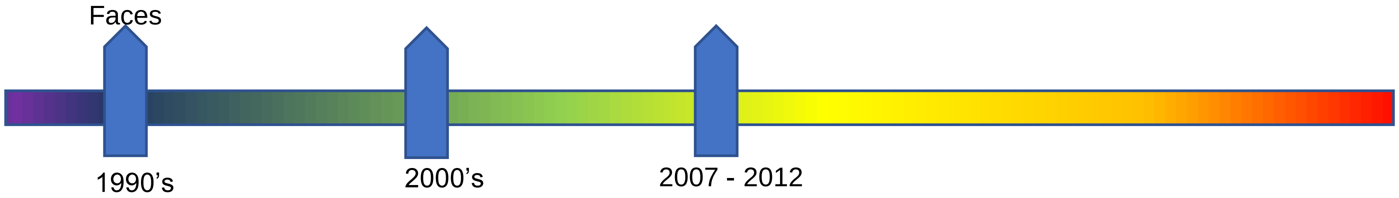
Applications: Semantic Segmentation



PASCAL VOC

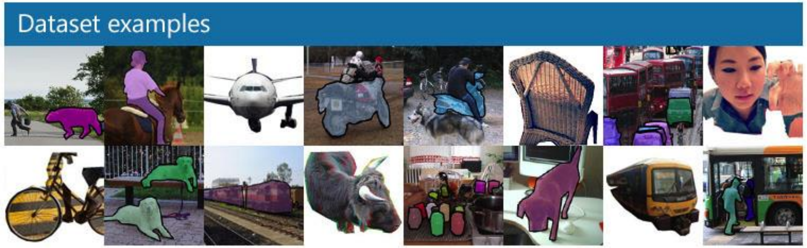
- 20 categories
- 10K images
- Large pose variations, heavy occlusions
- Generic scenes
- Cleaned up performance metric

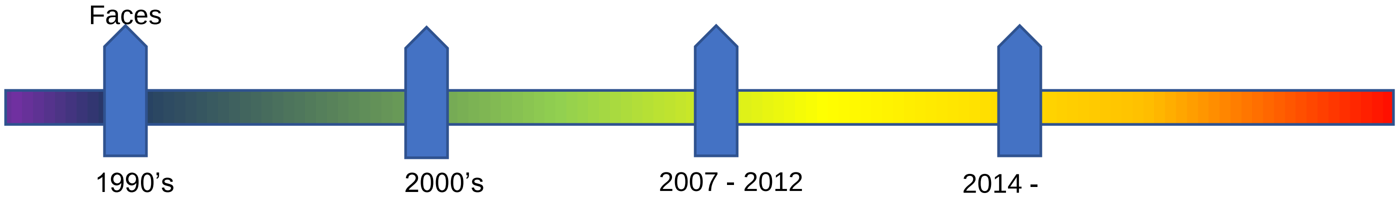




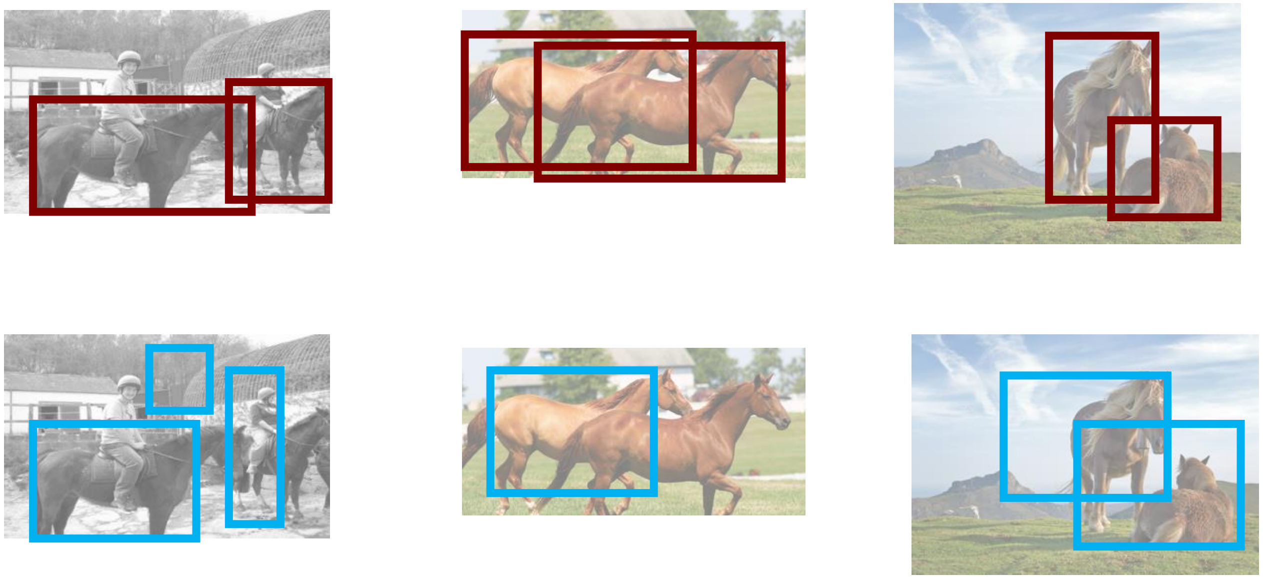
Coco

- 80 diverse categories
- 100K images
- Heavy occlusions, many objects per image, large scale variations

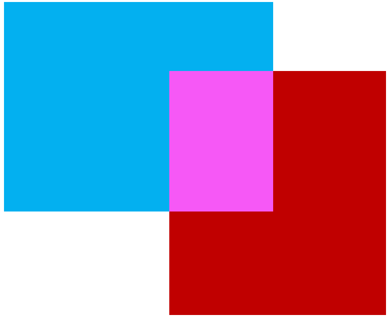




Evaluation metric



Matching detections to ground truth



$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

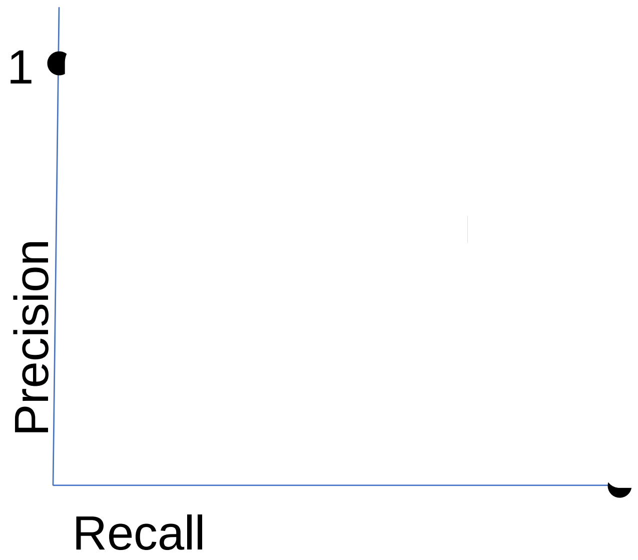
Matching detections to ground truth

- Match detection to most similar ground truth
  - highest IoU
- If IoU > 50%, mark as correct
- If multiple detections map to same ground truth, mark only one as correct
- **Precision** = #correct detections / total detections
- **Recall** = #ground truth with matched detections / total ground truth

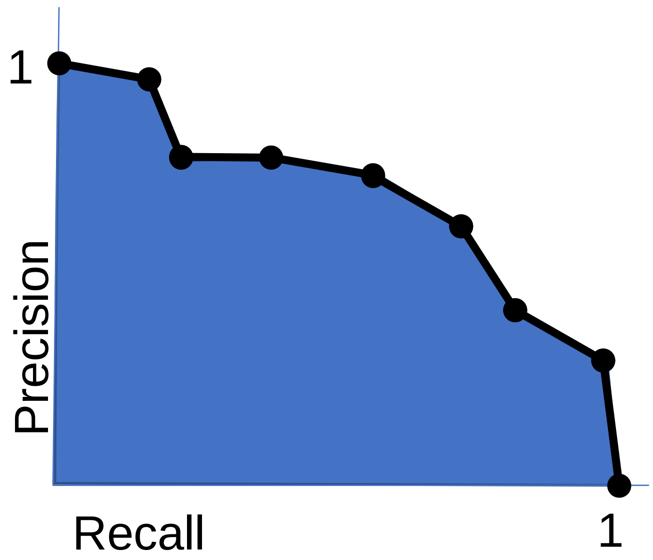
Tradeoff between precision and recall

- ML usually gives scores or probabilities, so we need to threshold
- Too low threshold → too many detections → low precision, high recall
- Too high threshold → too few detections → high precision, low recall
- Right tradeoff depends on application
  - Detecting cancer cells in tissue: need high recall
  - Detecting edible mushrooms in forest: need high precision

Average precision



Average precision



Average average precision

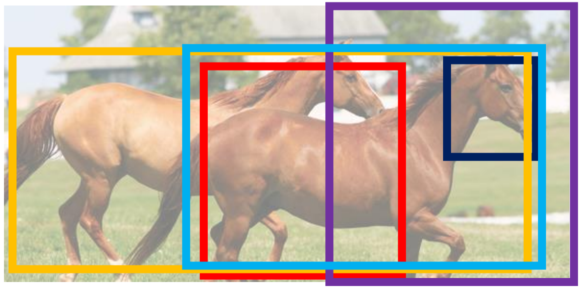
- AP marks detections with overlap > 50% as correct
- But may need better localization
- *Average* AP across multiple overlap thresholds
- Confusingly, still called average precision
- Introduced in COCO

Mean and category-wise AP

- Every category evaluated independently
- Typically report mean AP averaged over all categories
- Confusingly called “mean Average Precision”, or “mAP”

Why is detection hard(er)?

- Precise localization



Why is detection hard(er)?

- Much larger impact of pose

Why is detection hard(er)?

- Occlusion makes localization difficult

Why is detection hard(er)?

- Counting

Why is detection hard(er)?

- Small objects

Object detection and 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

Idea #1: Localization as Regression

Input: image

Neural Net

Output:
Box coordinates
(4 numbers)

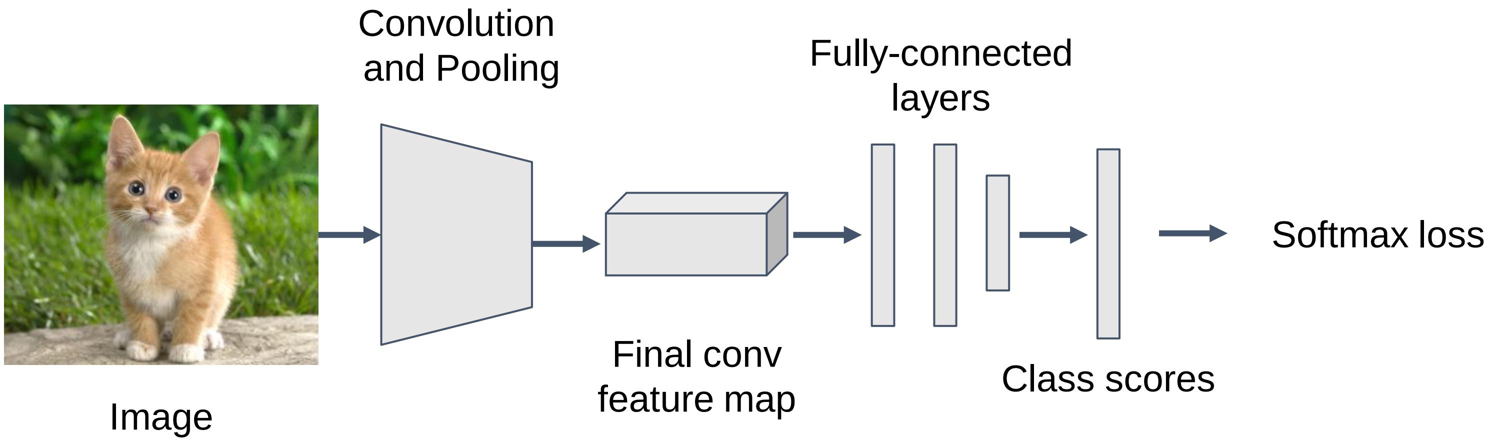
Correct output:
box coordinates
(4 numbers)

Loss:
L2 distance

Only one object,
simpler than detection

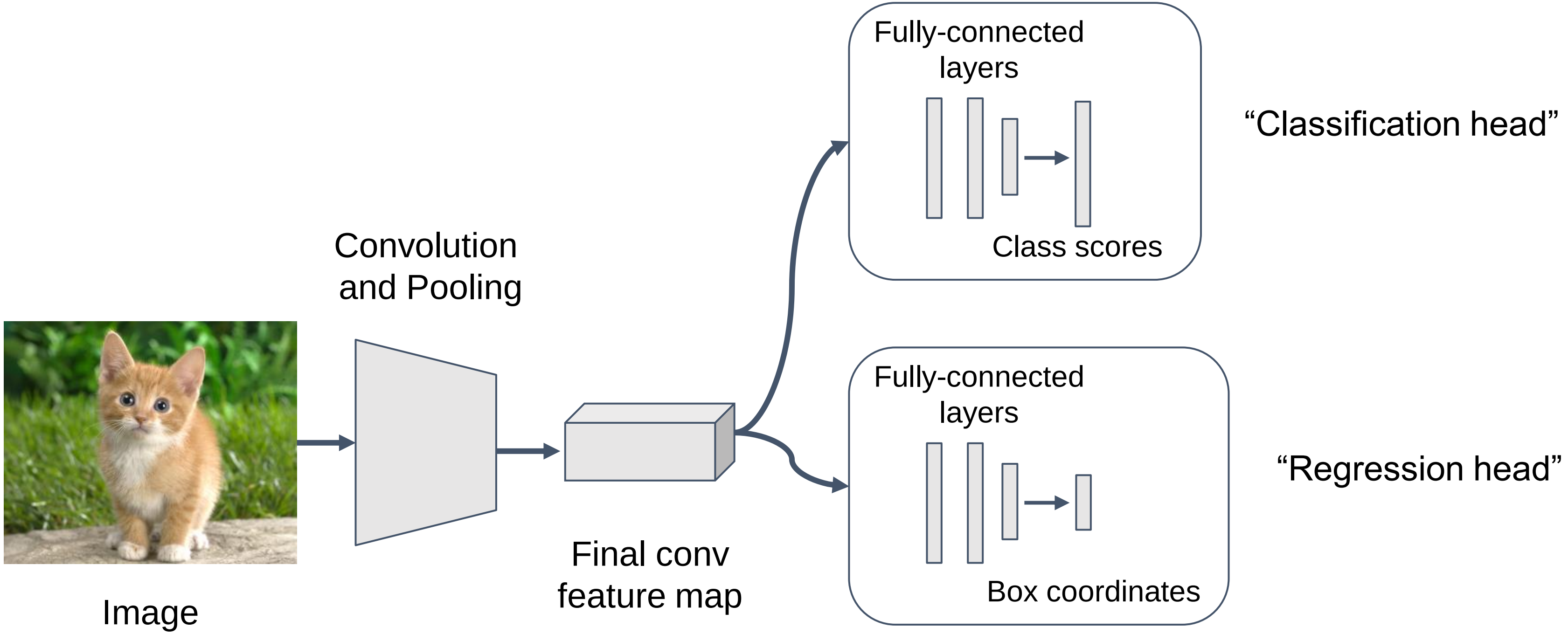
Simple Recipe for Classification + Localization

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



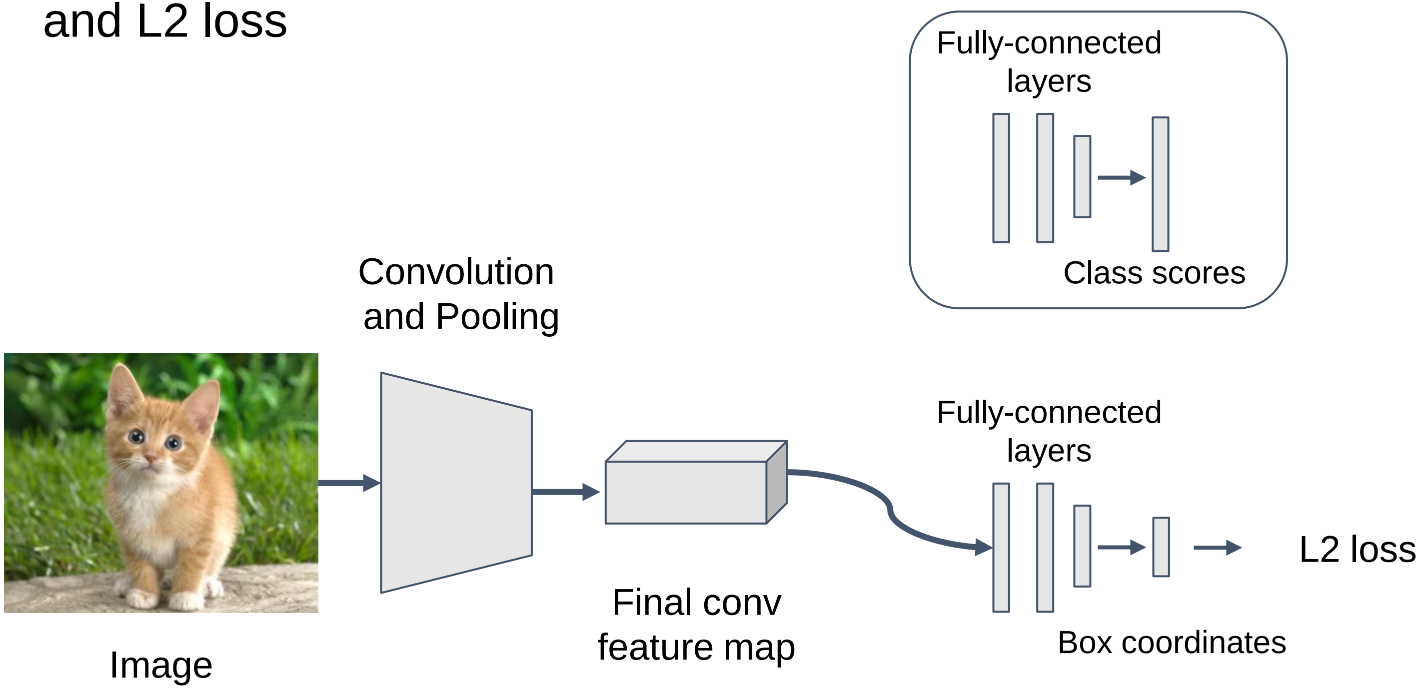
Simple Recipe for Classification + Localization

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



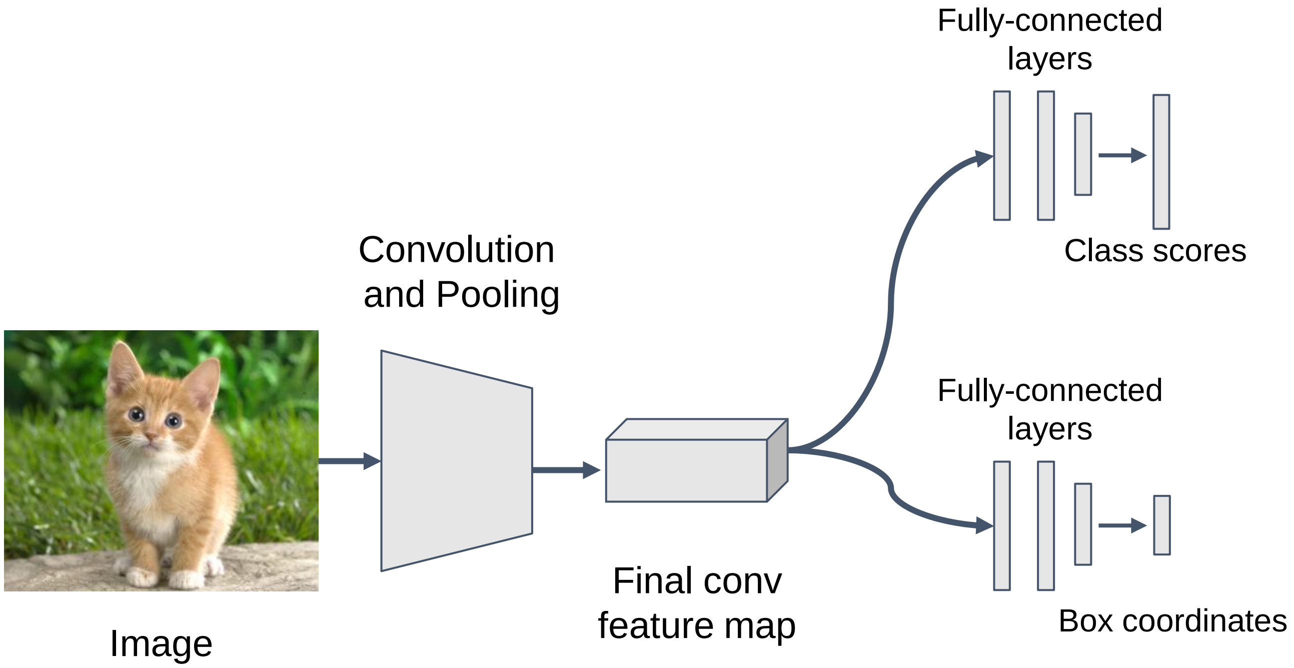
Simple Recipe for Classification + Localization

Step 3: Train the regression head only with stochastic gradient descent (SGD) and L2 loss



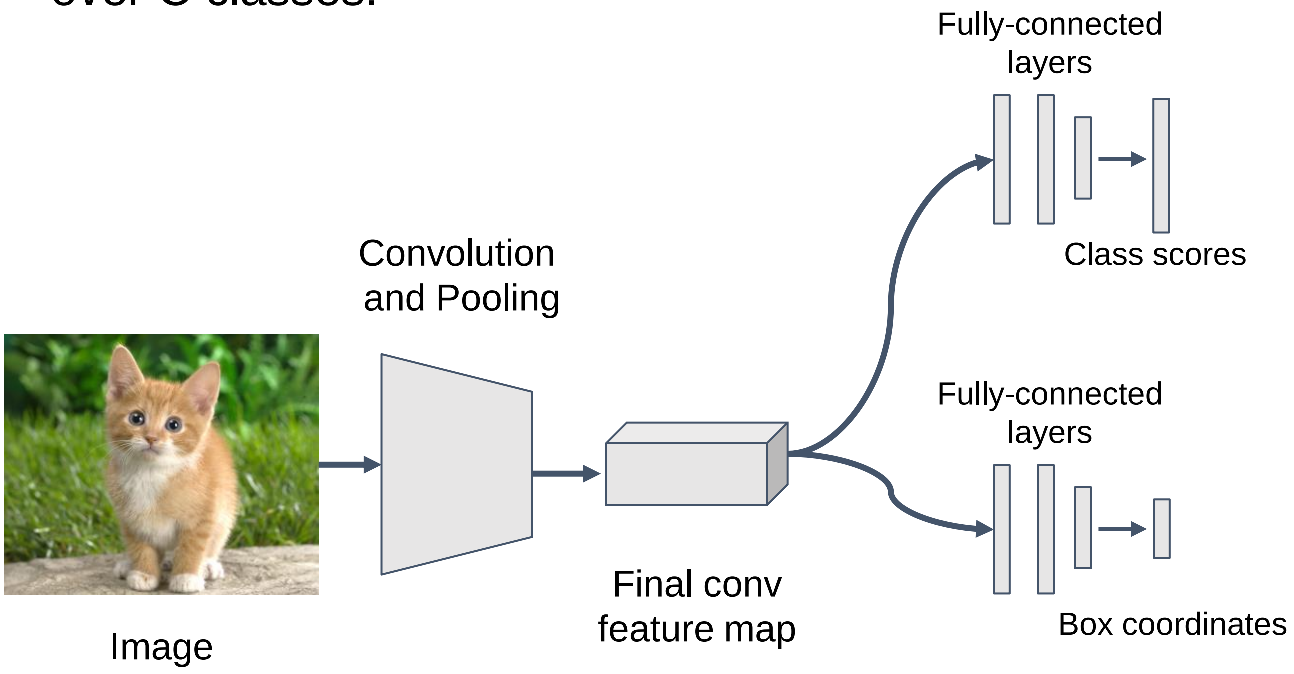
Simple Recipe for Classification + Localization

Step 4: At test time use both heads



Per-class vs class agnostic regression

Assume classification over C classes:

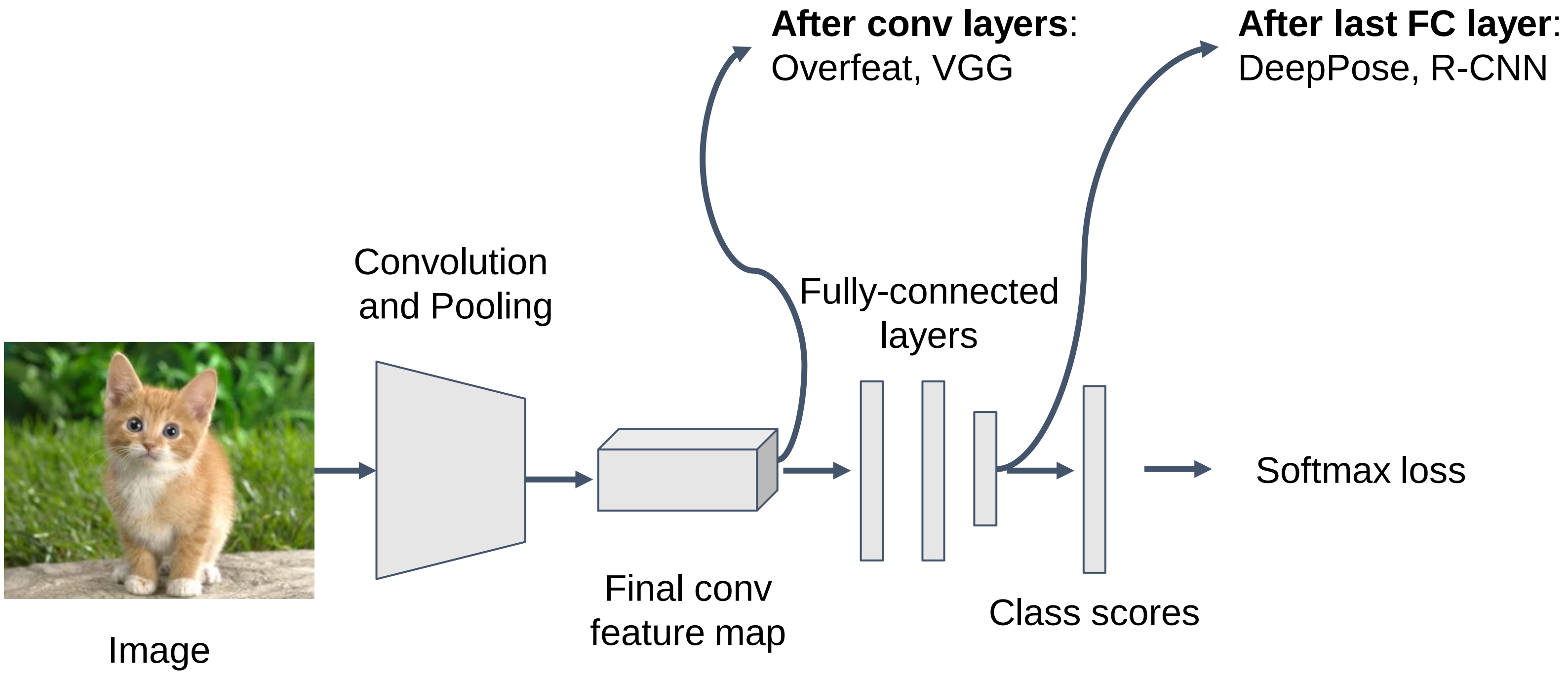


Classification head: C numbers (one per class)

Class agnostic: 4 numbers (one box)

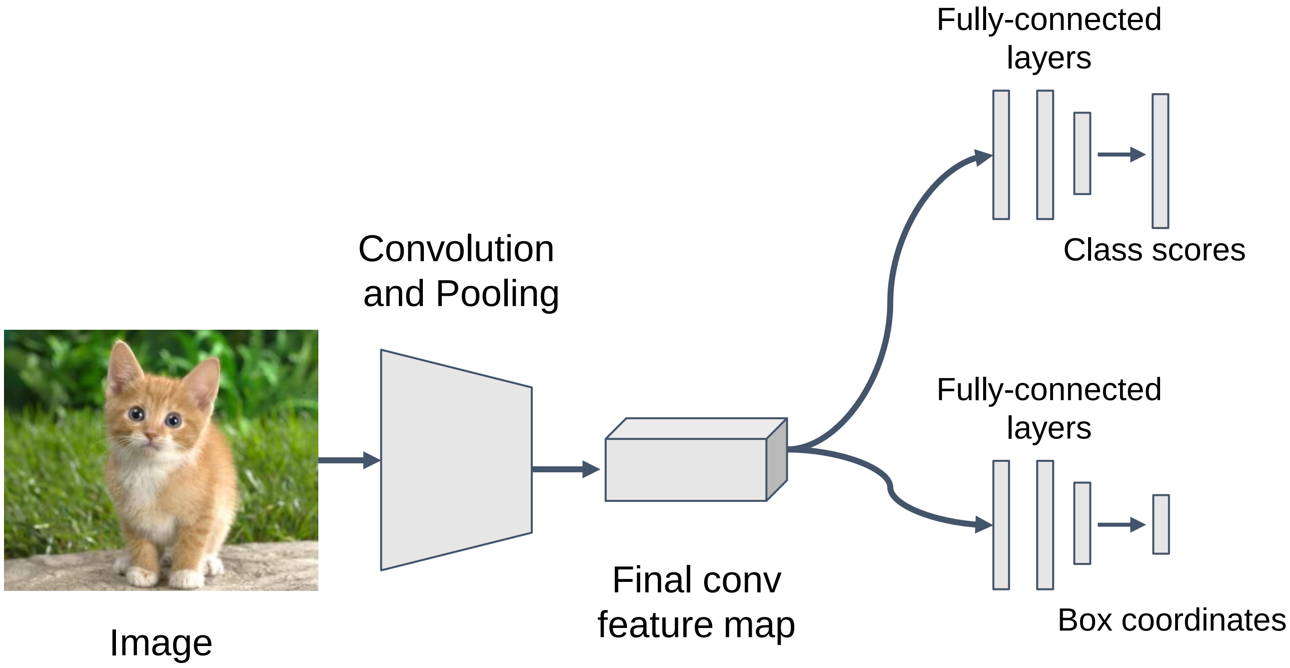
Class specific: C x 4 numbers (one box per class)

Where to attach the regression head?



Aside: Localizing multiple objects

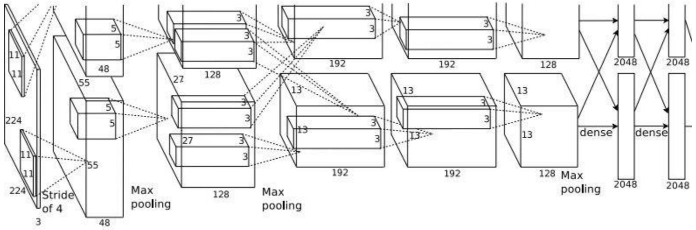
Want to localize **exactly** K objects in each image (e.g. whole cat, cat head, cat's left ear, cat 's ear for K=4)



K x 4 numbers
(one box per object)

Object Detection: Multiple Objects

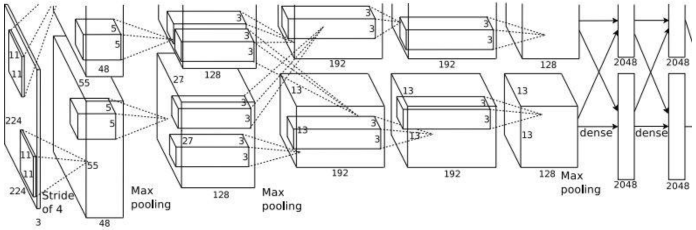
Apply a classifier to many different crops of the image; the classifier classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection: Multiple Objects

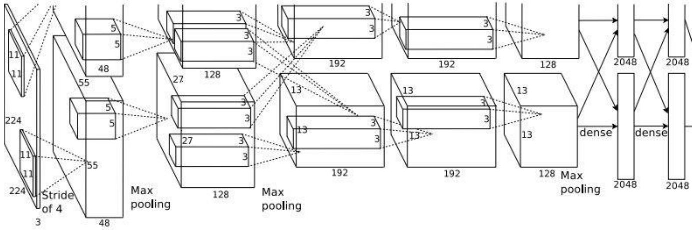
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

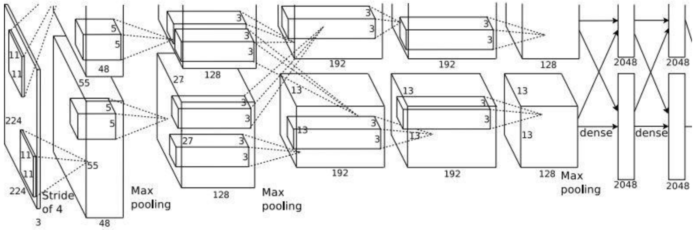
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

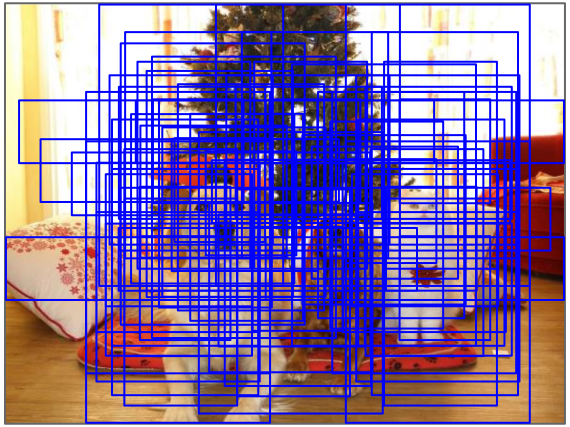


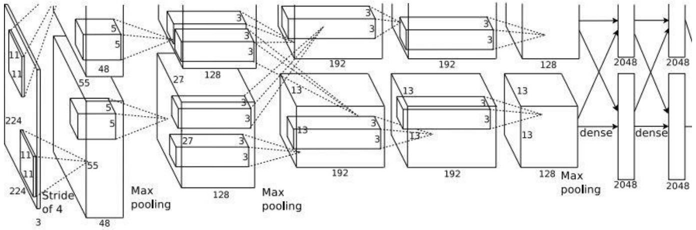
Dog? NO
Cat? YES
Background? NO

Q: What's the problem with this approach?

Object Detection: Multiple Objects

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background





Dog? NO
Cat? YES
Background? NO

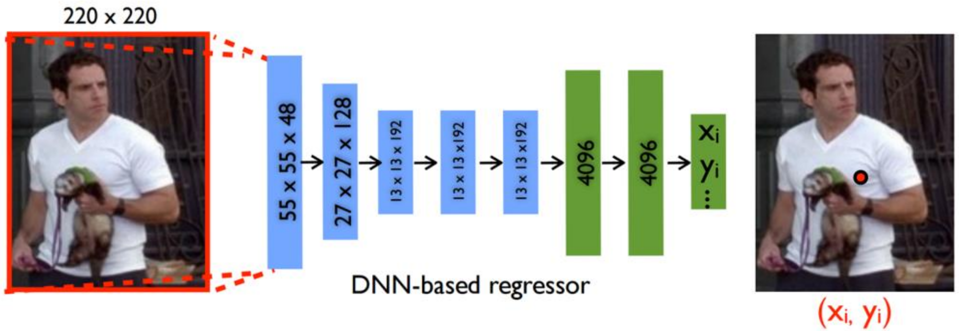
Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive!

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)



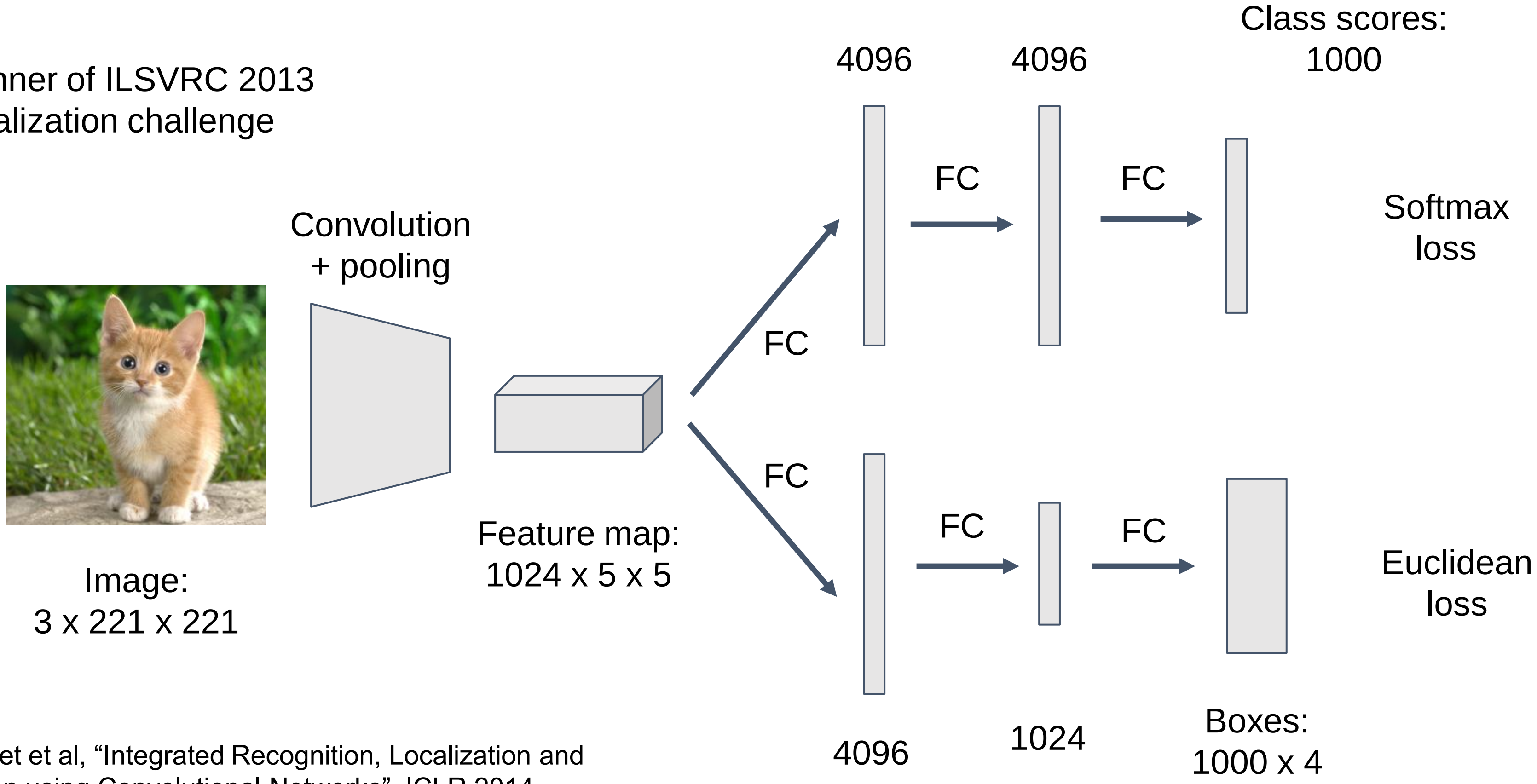
Toshev and Szegedy, “DeepPose: Human Pose Estimation via Deep Neural Networks”, CVPR 2014

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

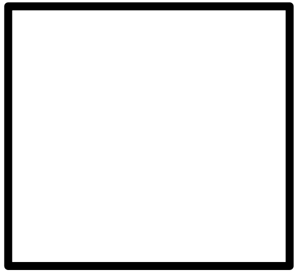
Sliding Window: Overfeat

Winner of ILSVRC 2013
localization challenge



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

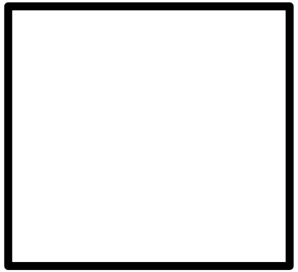
Sliding Window: Overfeat



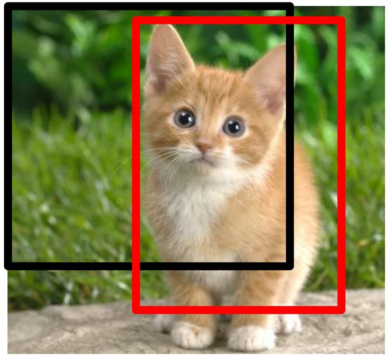
Network input:
3 x 221 x 221

Larger image:
3 x 257 x 257

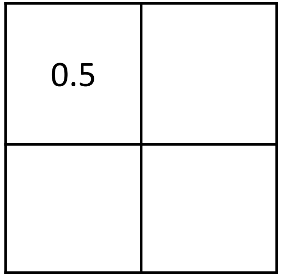
Sliding Window: Overfeat



Network input:
3 x 221 x 221

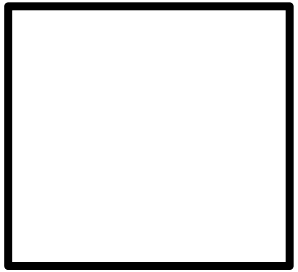


Larger image:
3 x 257 x 257

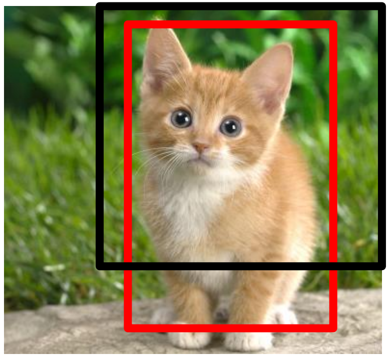


Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221

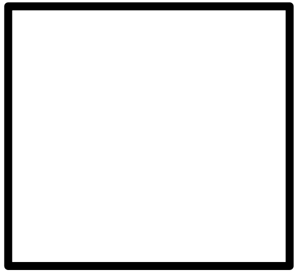


Larger image:
3 x 257 x 257

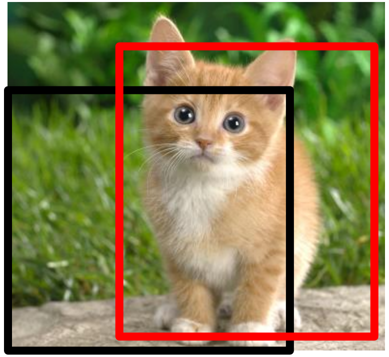
0.5	0.75

Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221



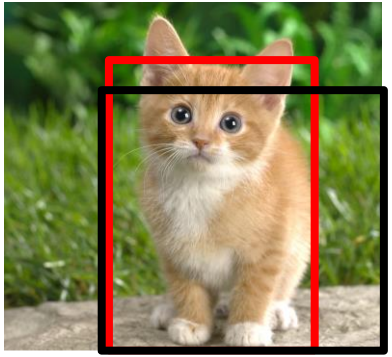
Larger image:
3 x 257 x 257

0.5	0.75
0.6	

Classification scores:
P(cat)

Sliding Window: Overfeat

Network input:
3 x 221 x 221

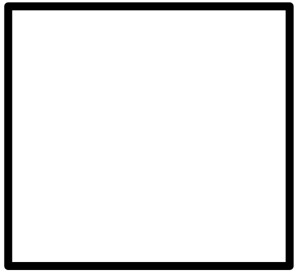


Larger image:
3 x 257 x 257

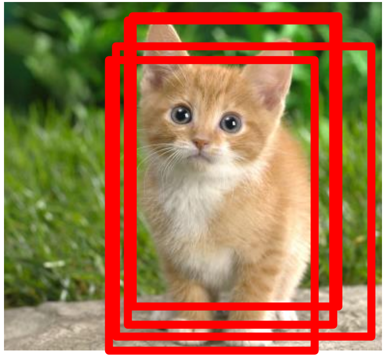
0.5	0.75
0.6	0.8

Classification scores:
P(cat)

Sliding Window: Overfeat



Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

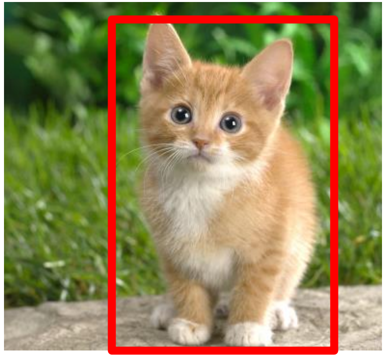
0.5	0.75
0.6	0.8

Classification scores:
P(cat)

Sliding Window: Overfeat

Greedily merge boxes and scores (details in paper)

Network input:
3 x 221 x 221



Larger image:
3 x 257 x 257

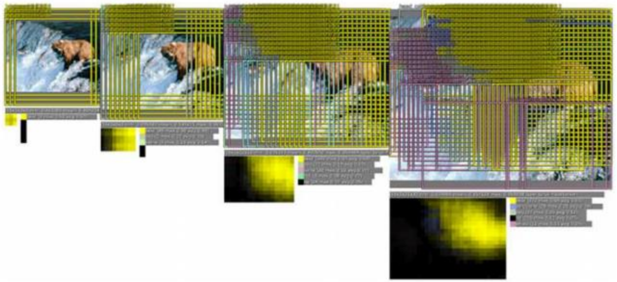
0.8

Classification score:
P(cat)

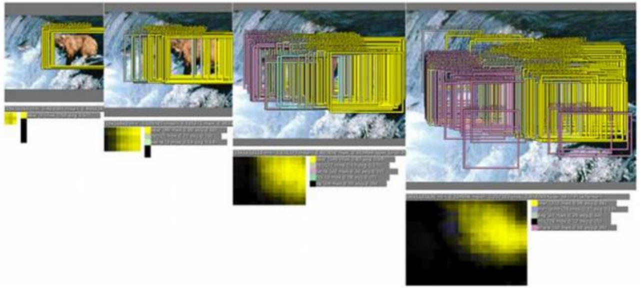
Sliding Window: Overfeat

In practice use many sliding window locations and multiple scales

Window positions + score maps



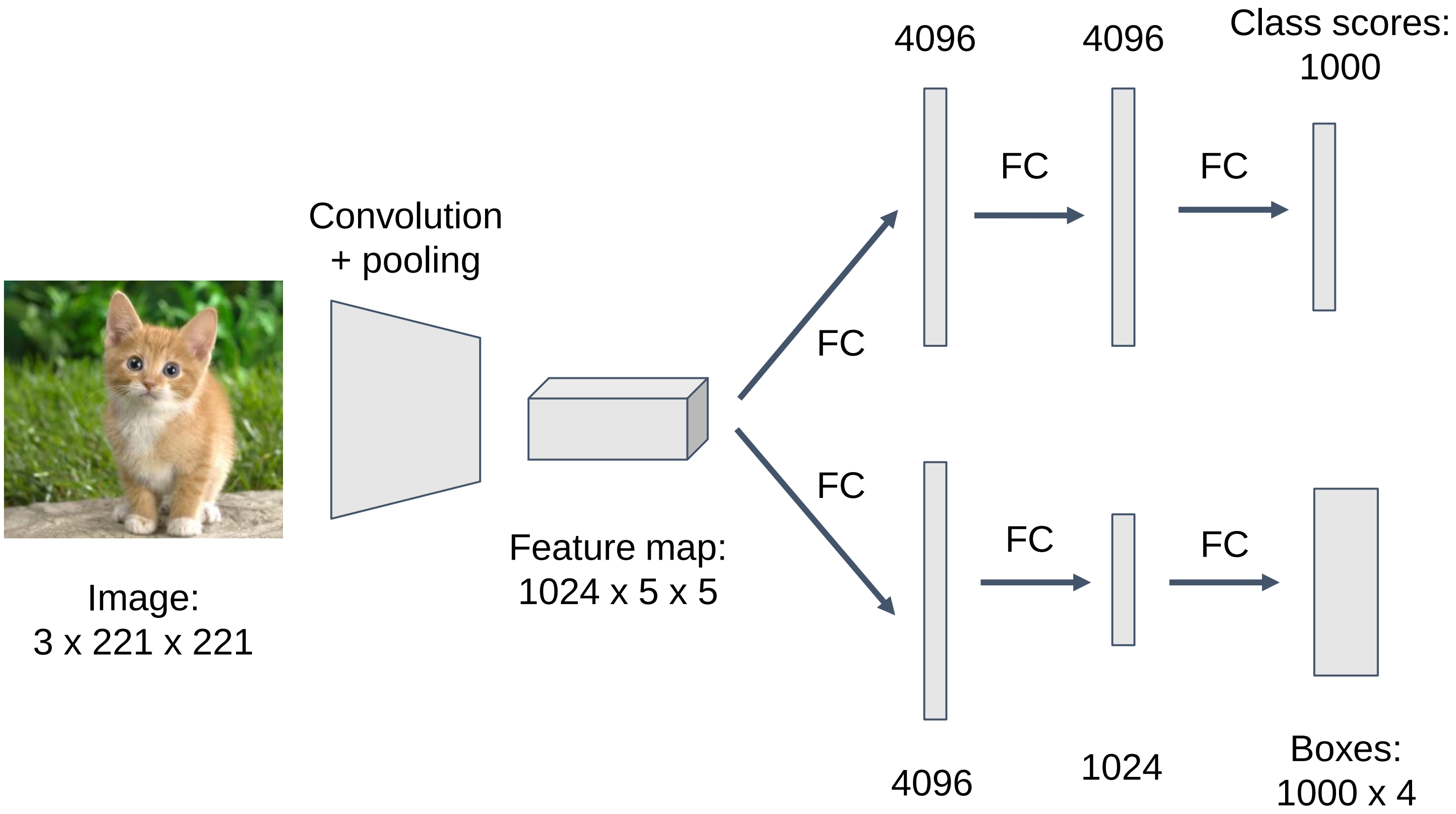
Box regression outputs



Final Predictions

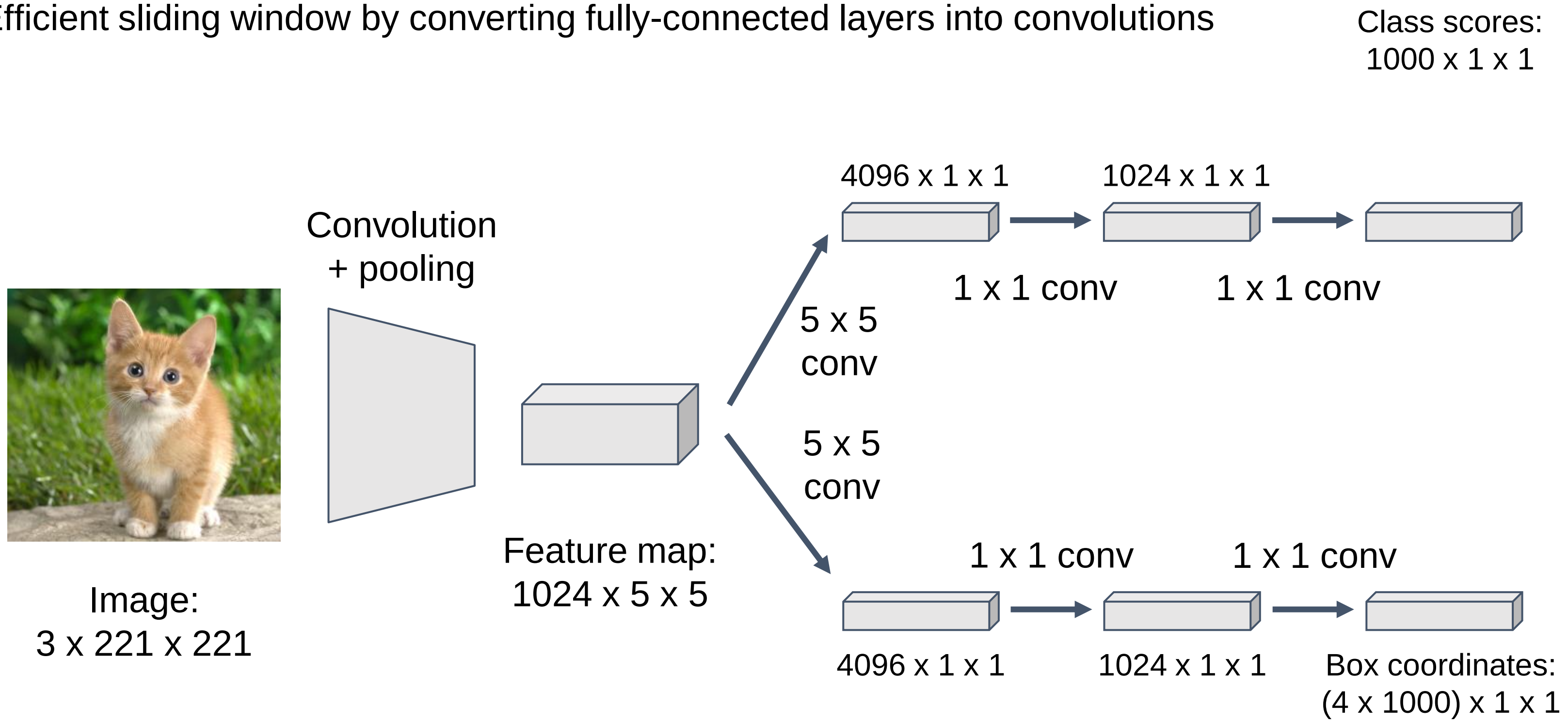
Sermanet et al, “Integrated Recognition, Localization and Detection using Convolutional Networks”, ICLR 2014

Efficient Sliding Window: Overfeat



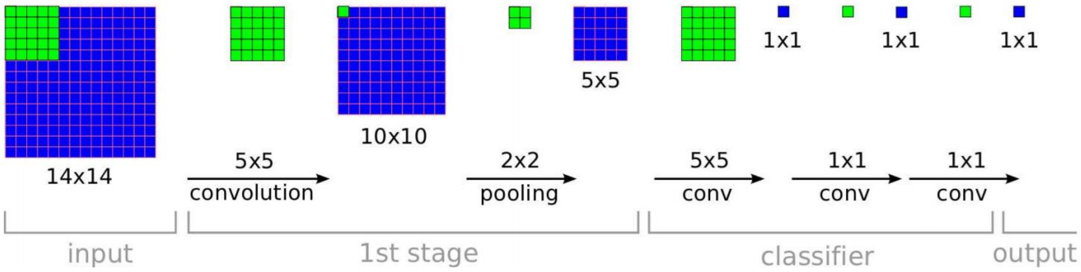
Efficient Sliding Window: Overfeat

Efficient sliding window by converting fully-connected layers into convolutions

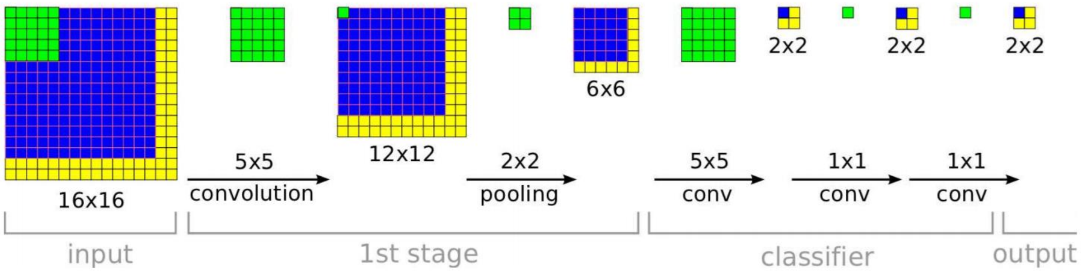


Efficient Sliding Window: Overfeat

Training time: Small image, 1 x 1 classifier output

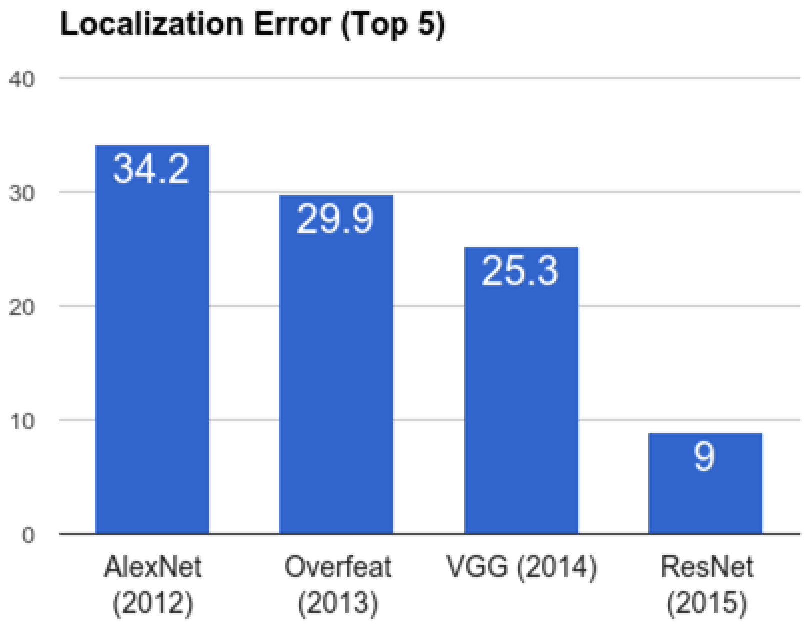


Test time: Larger image, 2 x 2 classifier output, only extra compute at yellow regions



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

ImageNet Classification + Localization



AlexNet: Localization method not published

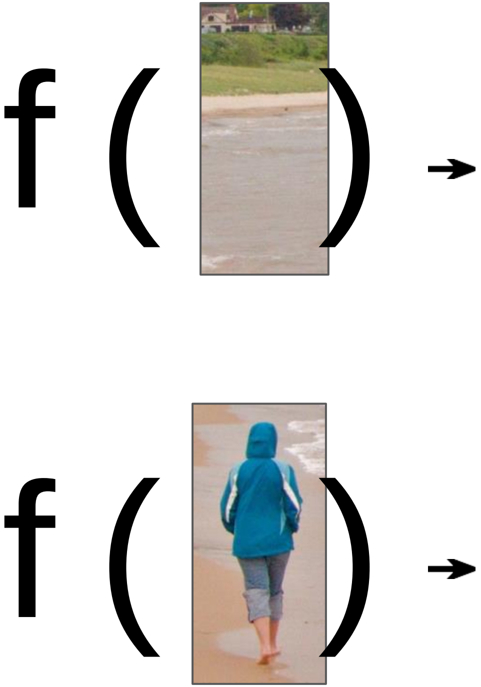
Overfeat: Multiscale convolutional regression with box merging

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

ResNet: Different localization method (Region Proposal Network - RPN) and much deeper features

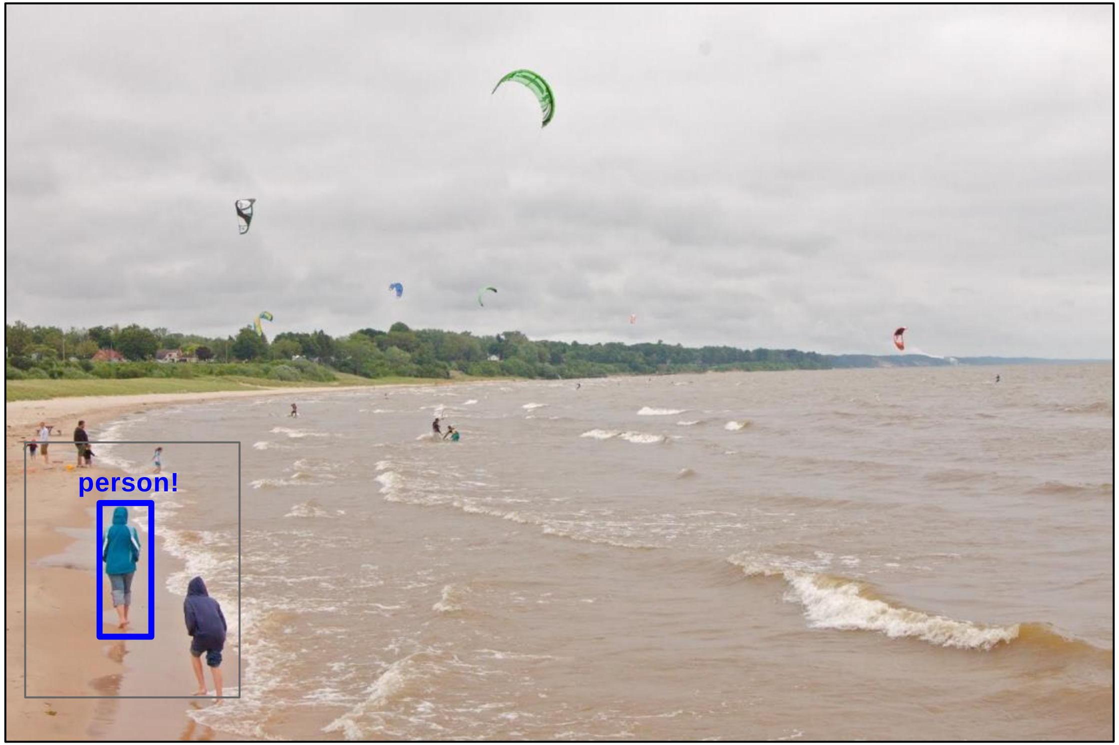
“Sliding Window” Detection

"Sliding Window" Detection



Compute within-region features, then classify

“Sliding Window” Detection



f ()

f ()

Typical to enlarge region to include some “context”

Sliding window placement

Slide over *fine grid*
in x, y, scale, aspect ratio

Slow and Accurate

Slide over *coarse grid*
in x, y, scale, aspect ratio

Fast and Not-so-accurate

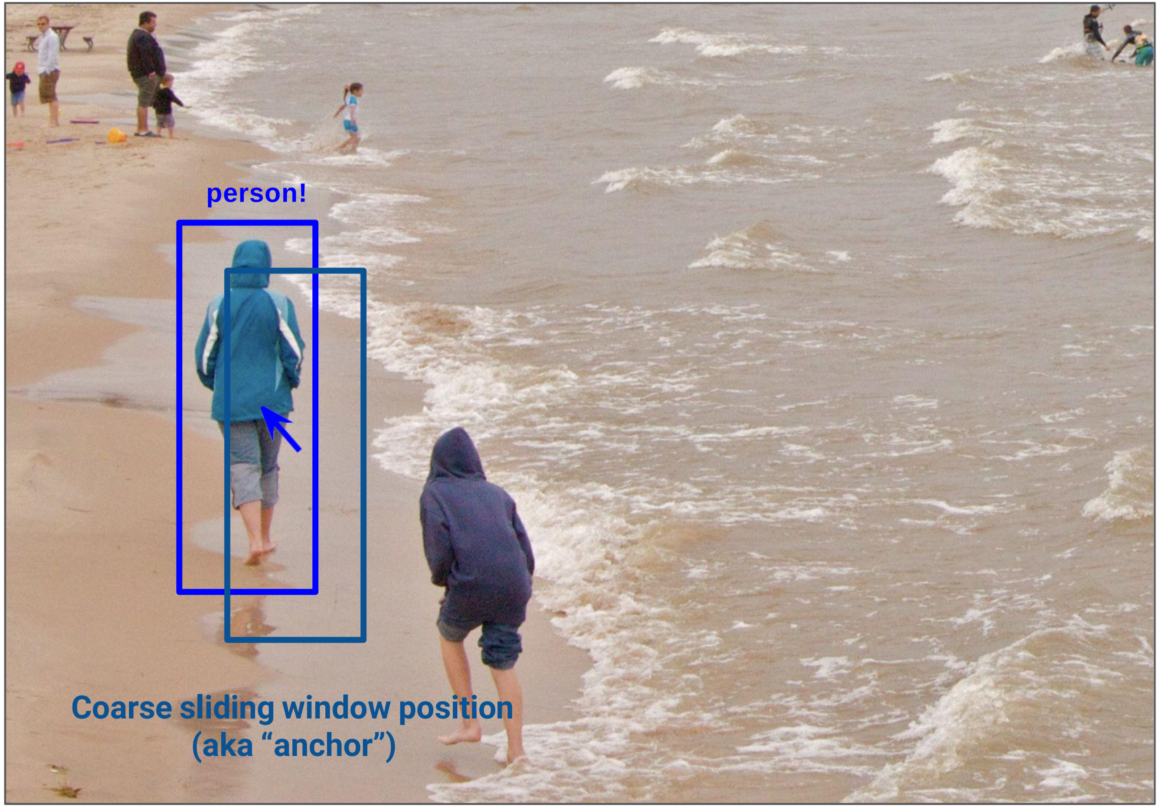
(... or can it be?)

Bounding Box Regression

Idea:

Also predict continuous offset from anchor to “snap” onto object

Bounding Box Regression



Idea:

Also predict continuous offset from anchor to “snap” onto object

Typical Training Objective

Common to use other location losses here...

Per-anchor Loss:

$$L(\text{anchor } \mathbf{a}) = \alpha * \delta(\mathbf{a} \text{ has matching groundtruth}) * L_2(\mathbf{t}^{loc}, W^{loc} \cdot \mathbf{v_{ij}})$$

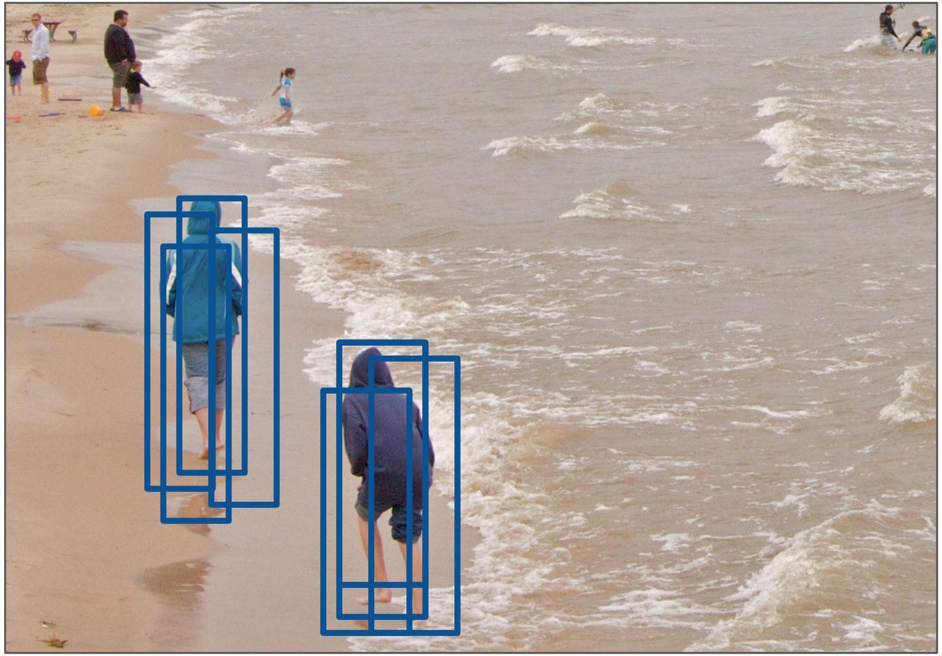
$$+ \quad \beta * \text{SoftMaxCrossEntropy}(\mathbf{t}^{cls}, W^{cls} \cdot \mathbf{v_{ij}})$$

Total Loss: Average per-anchor loss over anchors

Challenge: Dealing with class imbalance (usually way more negative anchors (class 0) than positive anchors)

Solutions: Subsampling negative anchors, downweighting the loss contribution of negatives, hard mining, etc...

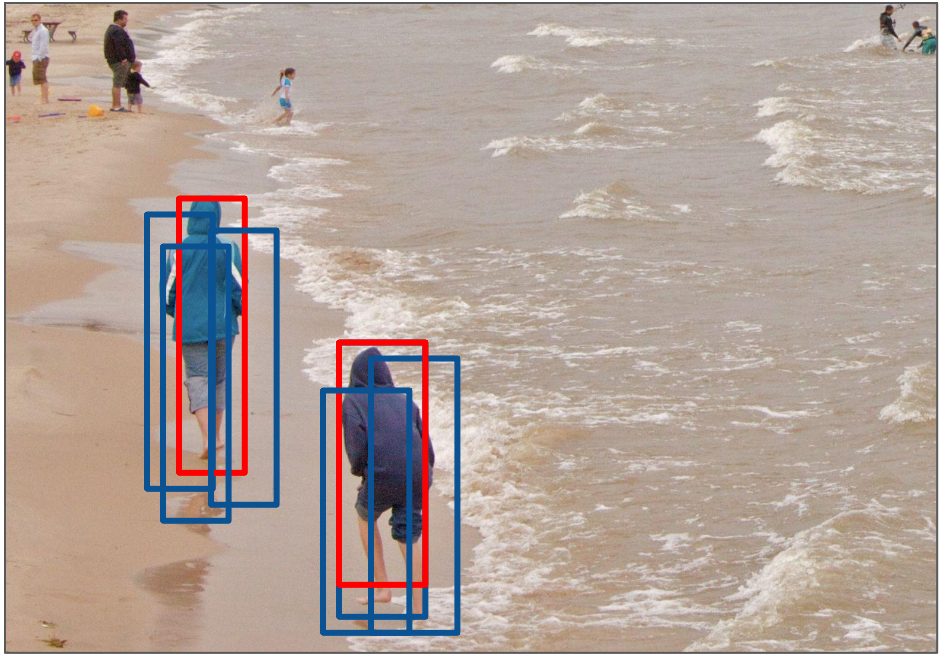
Dealing with multiple detections of the same object



Duplicate detection problem: Typically many anchors will detect the same underlying object and give slightly different boxes, with slightly different scores.

Solution: remove detections if they overlap too much with another higher scoring detection.

Non Max Suppression (NMS)



Algorithm:

1. **Sort detections in decreasing order with respect to score**
2. **Iterate through sorted detections:**
   Reject a detection if it overlaps with a previous (unrejected) detection with IOU greater than some threshold
3. **Return all unrejected detections**

Some shortcomings of NMS to remember:

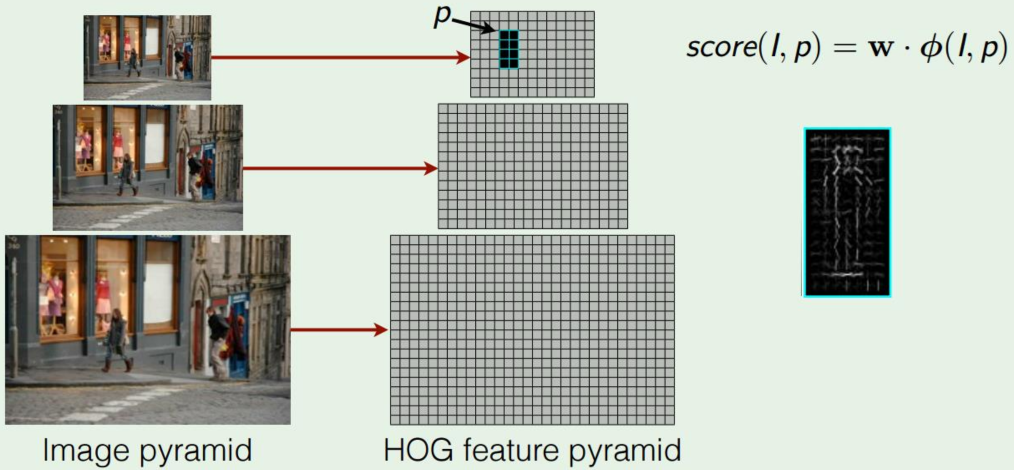
- **Imposes a hard limitation on how close objects can be in order to be detected**
- **Similar classes do not suppress each other**

Detection as Classification

Problem: Need to test many positions and scales

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

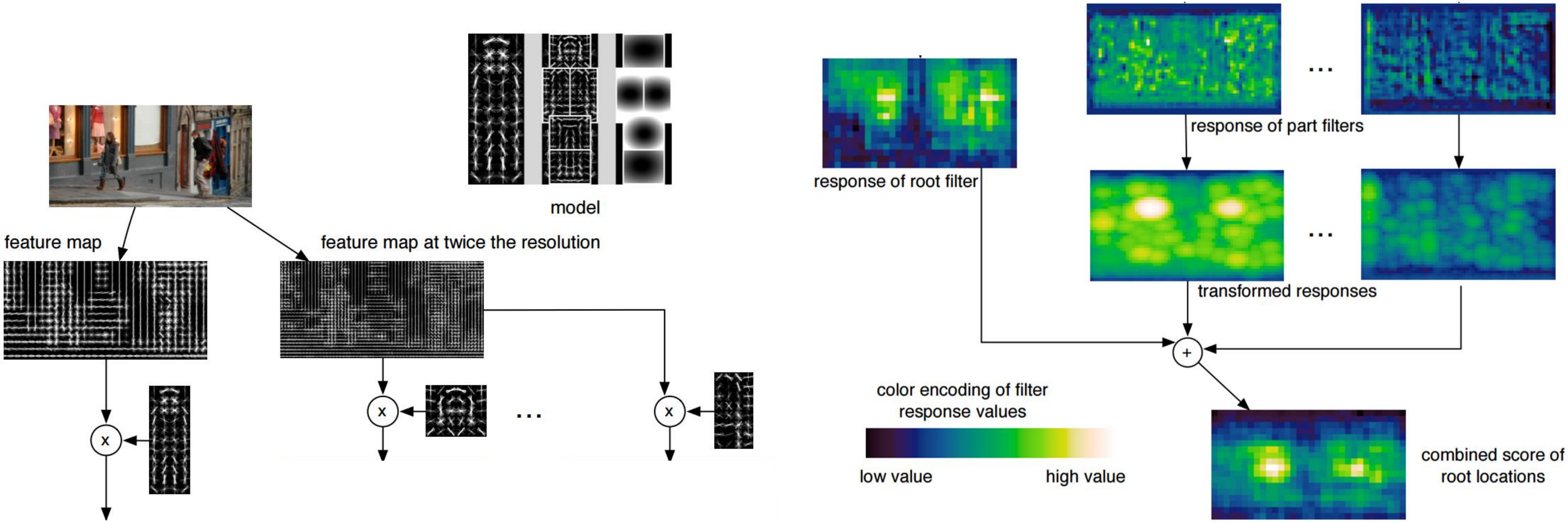
Histogram of Oriented Gradients



- Compute HOG of the whole image at multiple resolutions
- Score every subwindow of the feature pyramid
- Apply non-maxima suppression

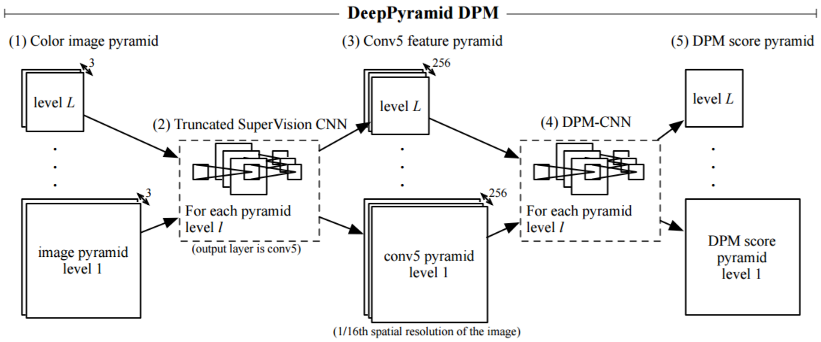
Dalal and Triggs, “Histograms of Oriented Gradients for Human Detection”, CVPR 2005
Slide credit: Ross Girshick

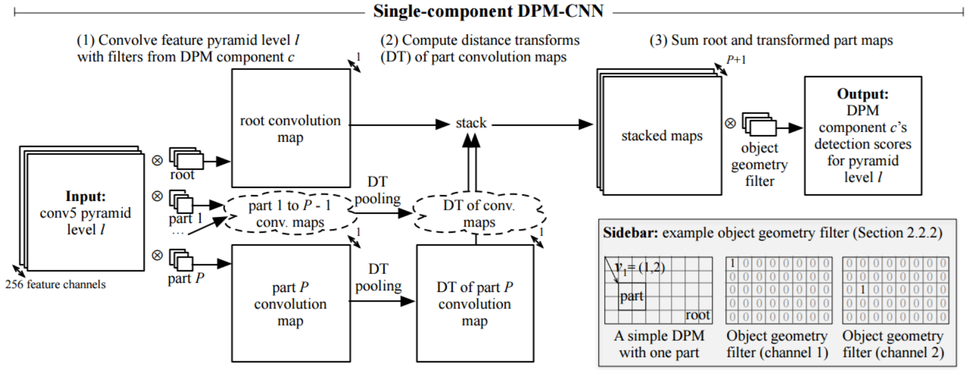
Deformable Parts Model (DPM)



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