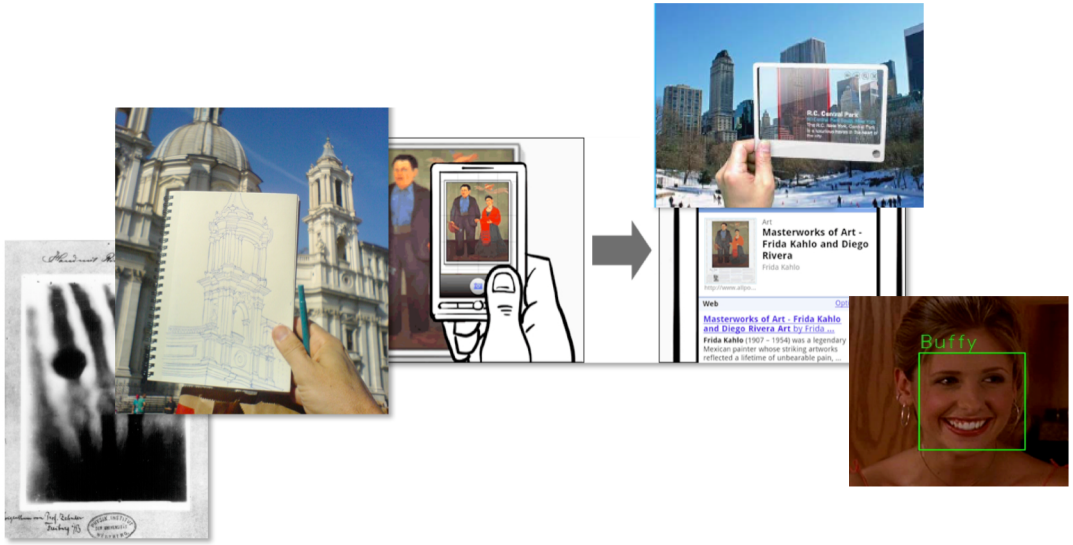



# Deep (7): Detection, Segmentation

Matthieu Cord LIP6 / SU

# Outline

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
4. Beyond ImageNet
    1. Fully Convolutional Networks (FCNs)
    2. Transfer
    3. **Detection**
    4. Segmentation

# Computer Vision Tasks

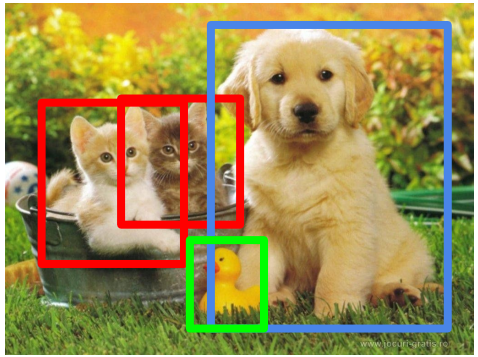
**Classification**

CAT

**Classification + Localization**

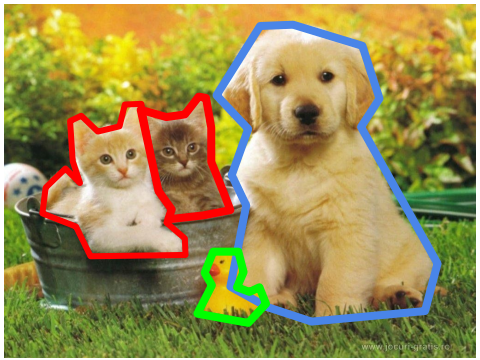
CAT

**Object Detection**



CAT, DOG, DUCK

**Instance Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

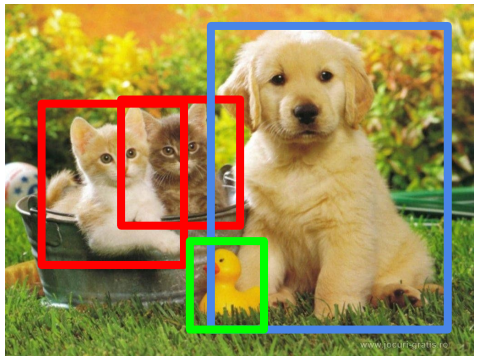
# Computer Vision Tasks

Classification

Classification + Localization

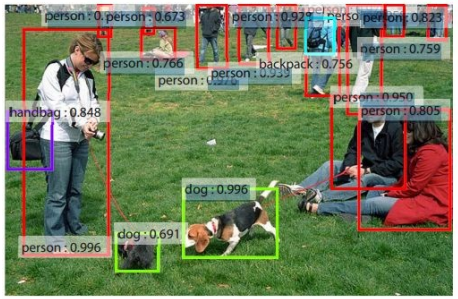
**Object Detection**

Instance Segmentation

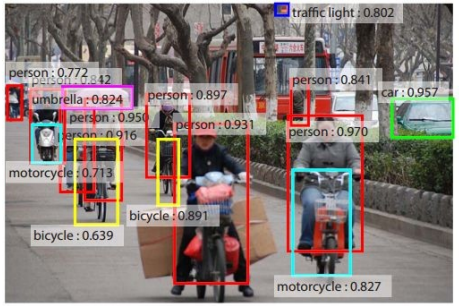


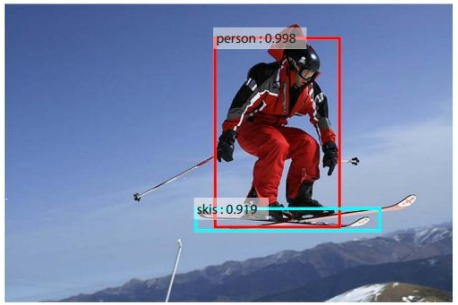
**Potentially different objects per image**
**Bounding boxes available in train**

# Detection


person : 0.673
person : 0.766
backpack : 0.756
person : 0.759
handbag : 0.848
person : 0.950
person : 0.805
dog : 0.996
dog : 0.691
person : 0.996


airplane : 0.997
person : 0.867

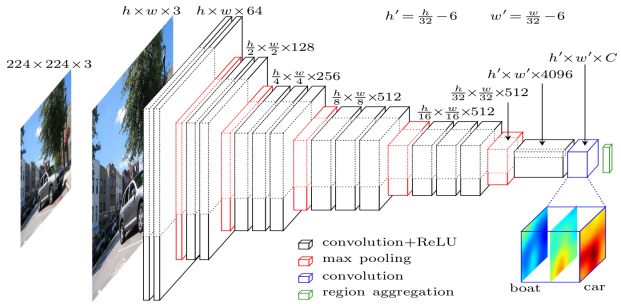

traffic light : 0.802
person : 0.772
umbrella : 0.824
person : 0.897
person : 0.841
car : 0.957
person : 0.950
person : 0.931
person : 0.970
motorcycle : 0.713
bicycle : 0.891
bicycle : 0.639
motorcycle : 0.827

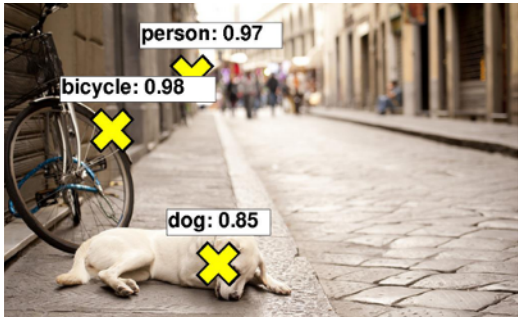

person : 0.998
skis : 0.919


clock : 0.982

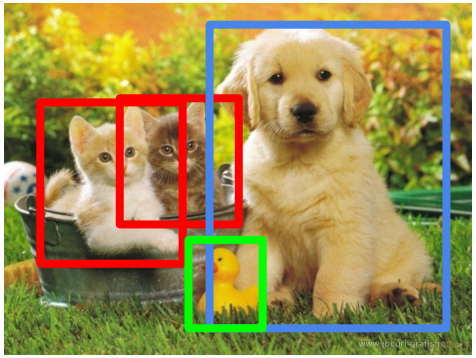

bed : 0.999

# Recap compa. With WSL for classif/localization vs. Detection



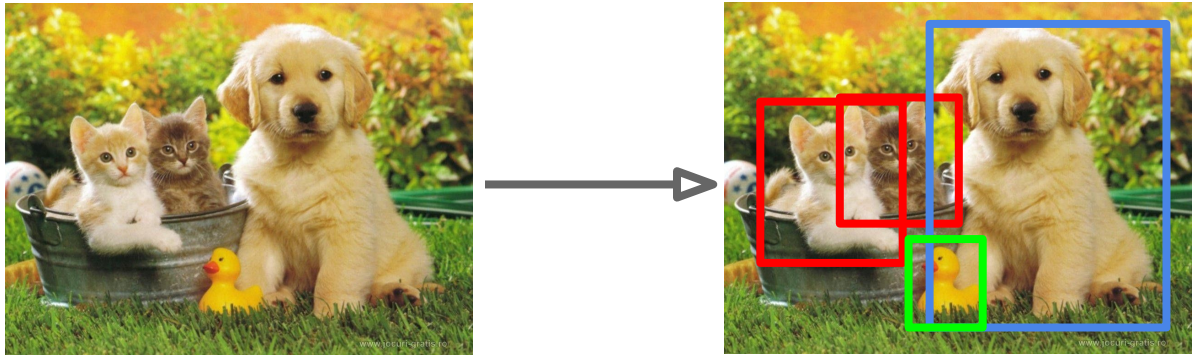

Classification
NO Box labels
ONLY global labels



Fully supervised detection
Box labels available in train

# Detection

Modeling as a regression problem?



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
=> Not easy to model

# Detection as Classification

Modeling detection as classification with a sliding window approach?

**CAT? NO**

**DOG? NO**

# Detection as Classification

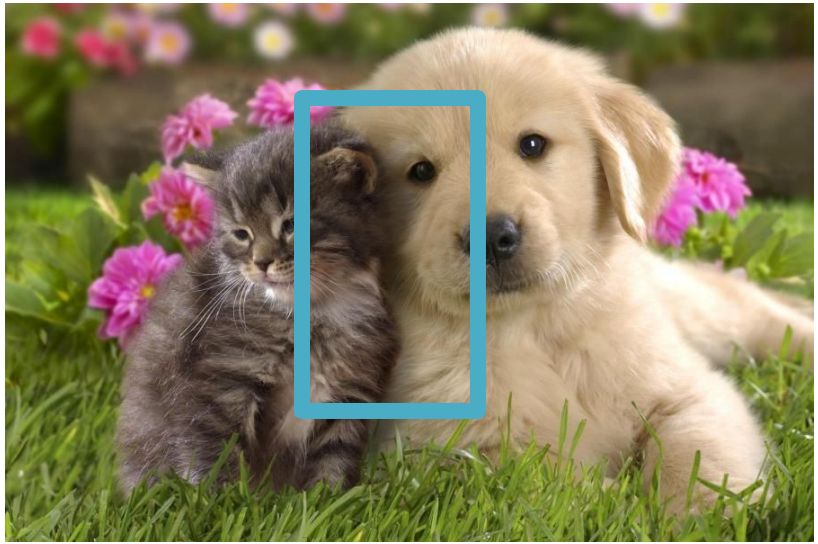
Modeling detection as classification with a sliding window approach?

**CAT? YES!**

**DOG? NO**

# Detection as Classification

Modeling detection as classification with a sliding window approach?

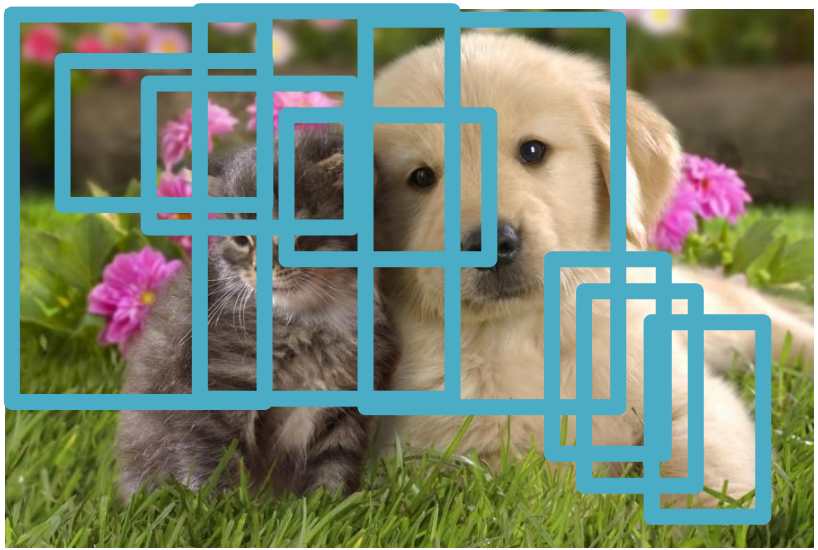


**CAT? NO**

**DOG? NO**

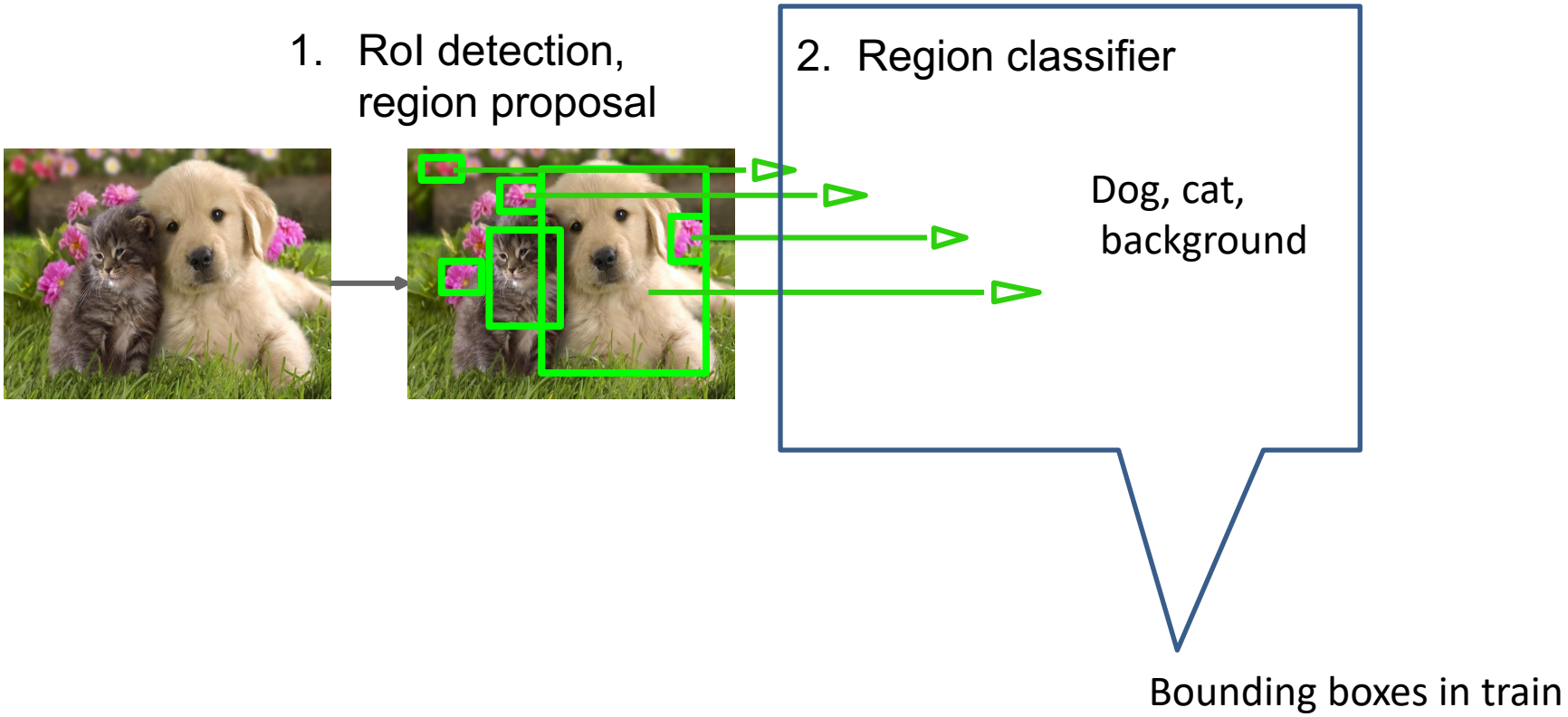
# Detection as Classification

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



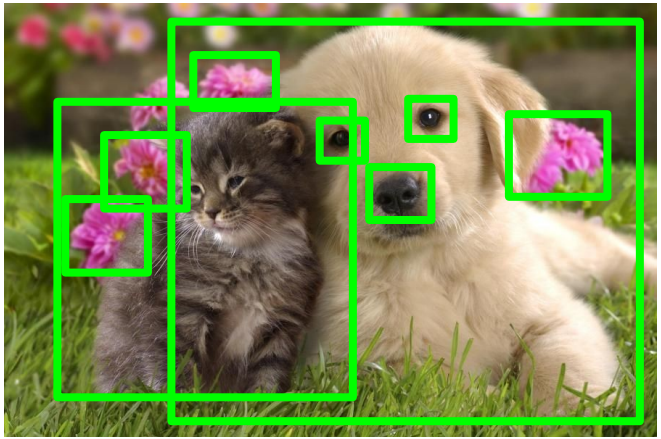
**Solution:** Only look at a tiny subset of possible positions

# Detection scheme



# Region Proposals

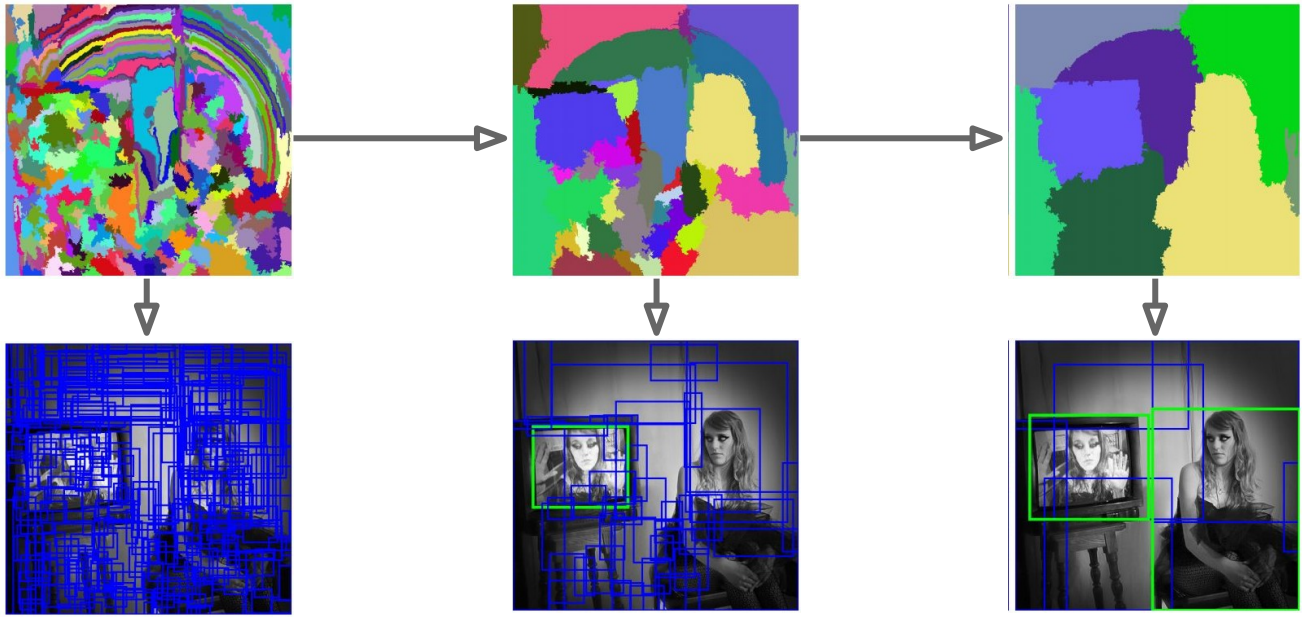
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



# Region Proposals: Selective Search

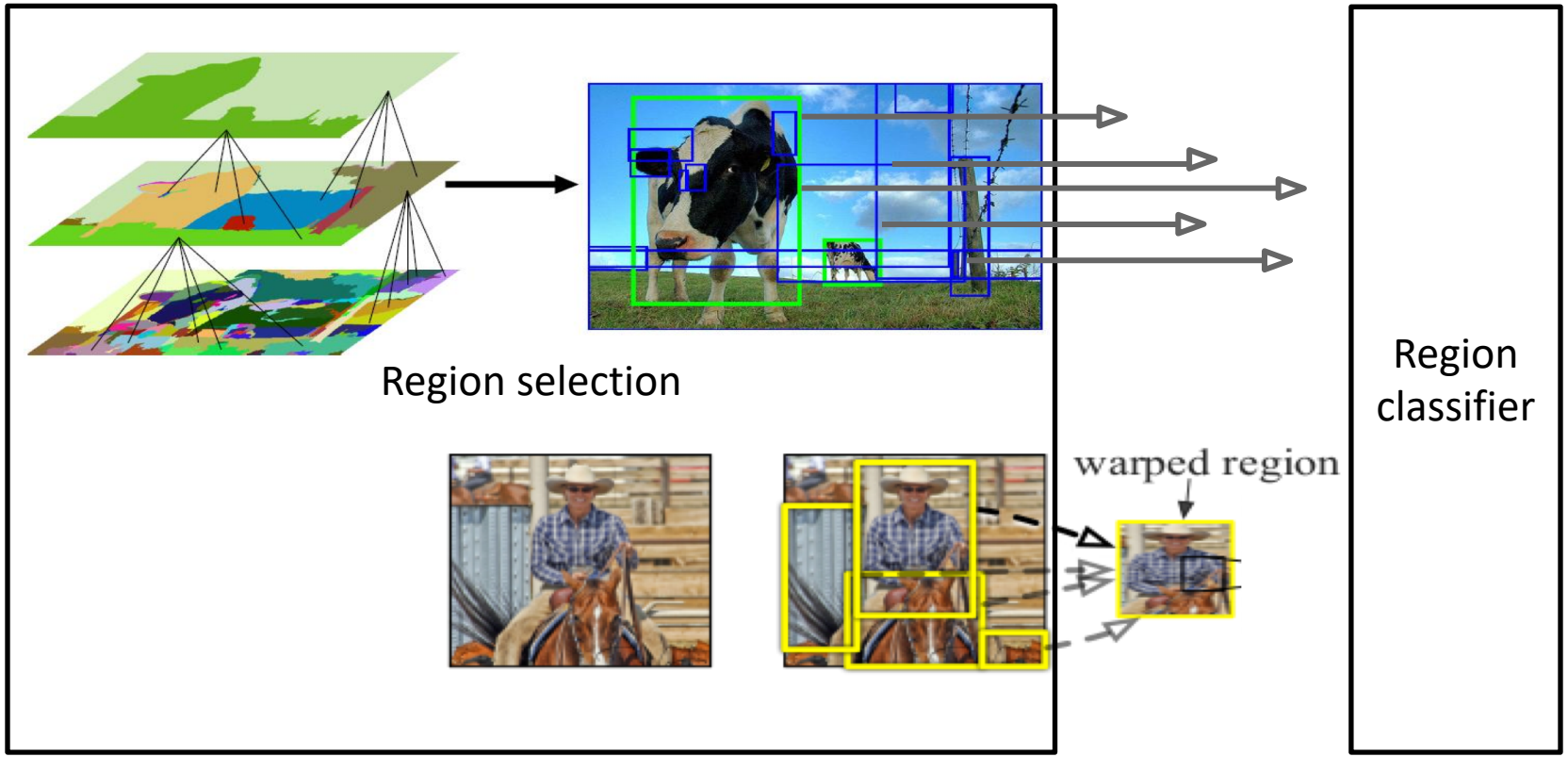
Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes



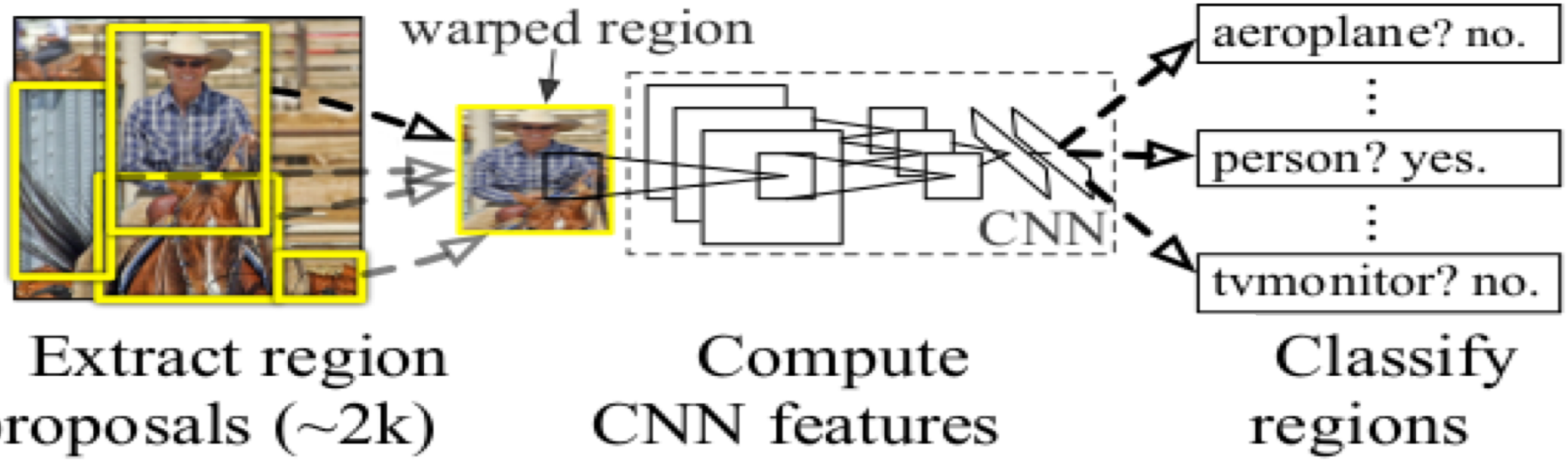
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013

# Detection scheme

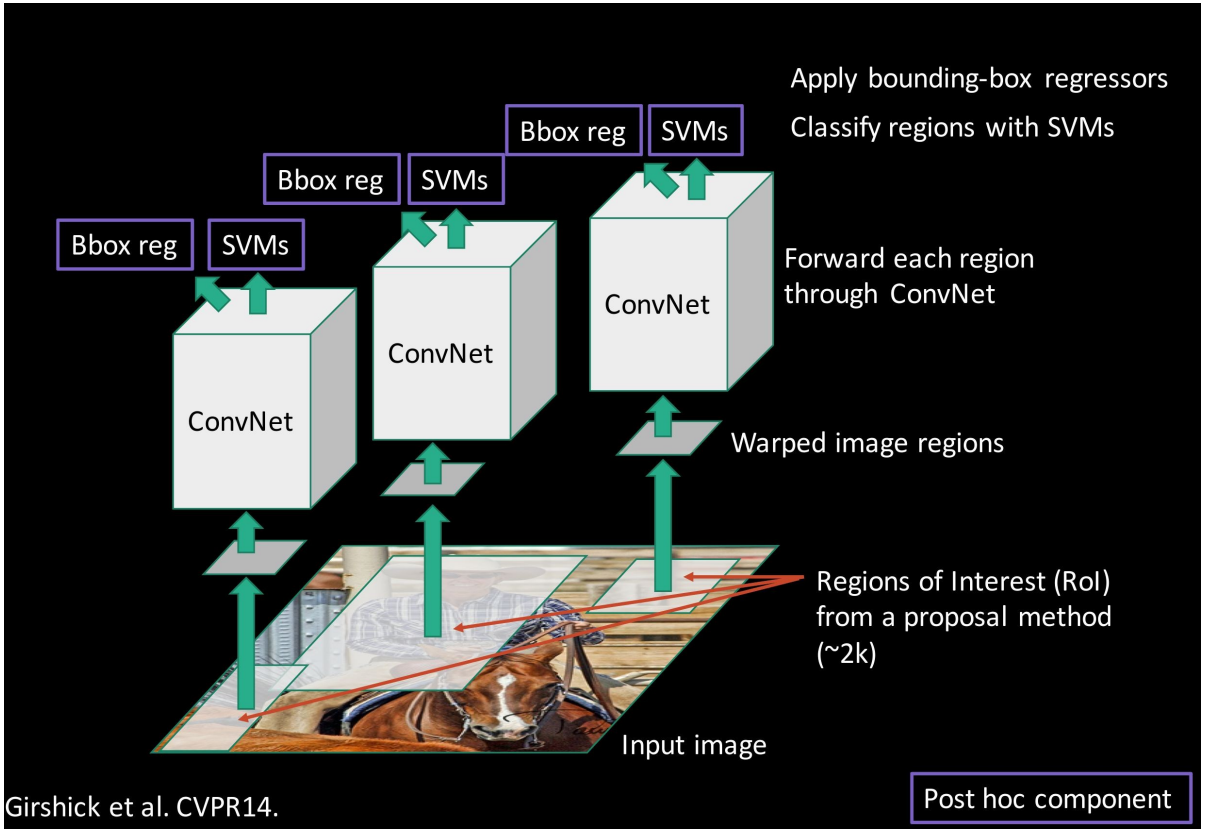


# Detection: R-CNN approach [CVPR 2014]

Region CNN (R-CNN) approach



# Putting it together: R-CNN

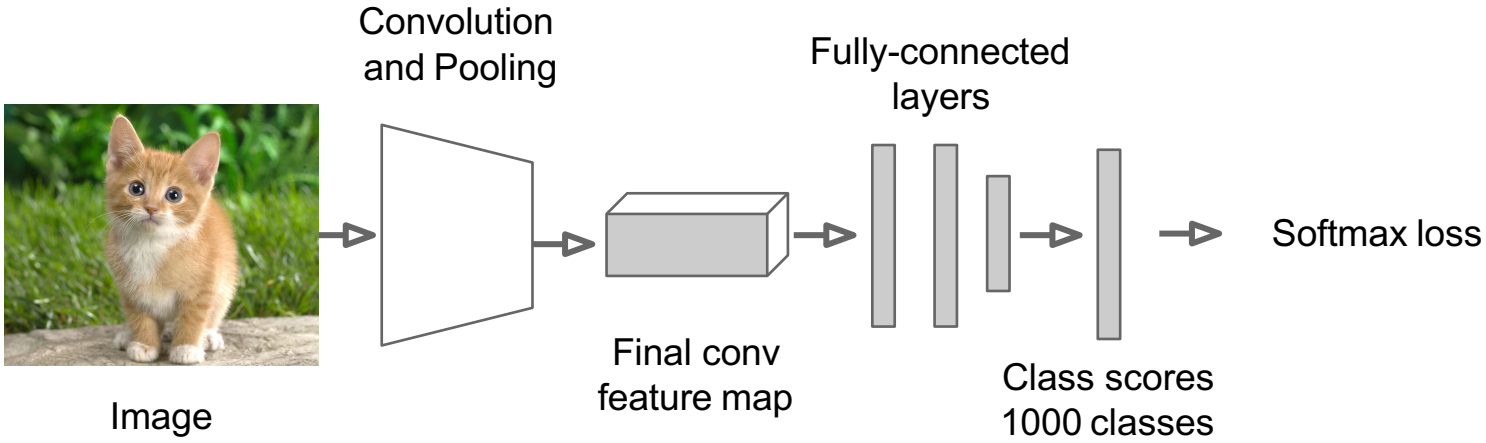



Girschick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014

Slide credit: Ross Girschick

# R-CNN Training

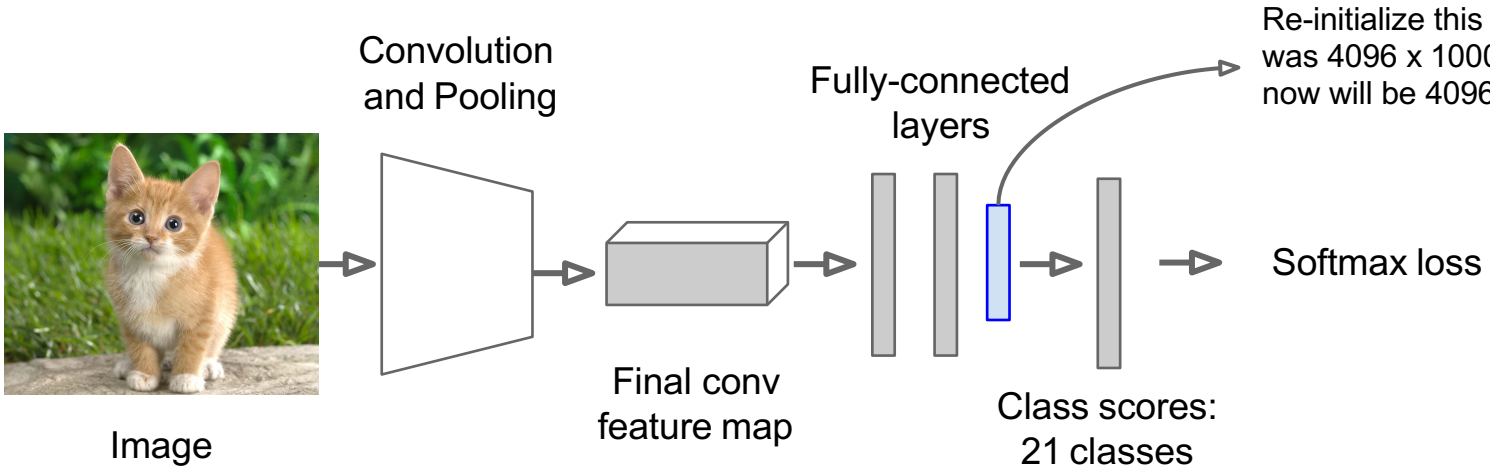
**Step 1**: Train (or download) a classification model for ImageNet (AlexNet)



# R-CNN Training

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

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

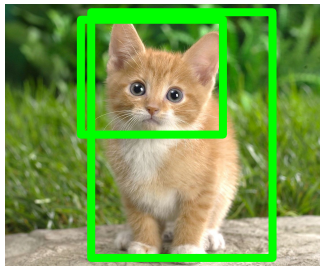


# R-CNN Training

**Step 3**: Extract features

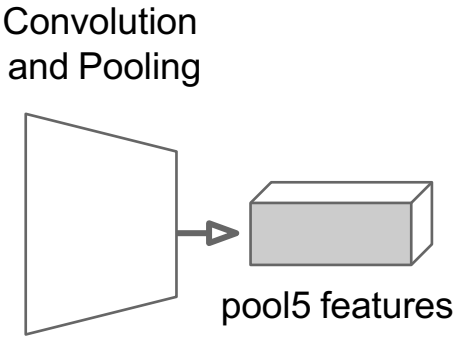
- Extract region proposals for all images
- For each region: warp to CNN input size, run forward through CNN, save pool5 features to disk
- Have a big hard drive: features are ~200GB for PASCAL dataset!

Image

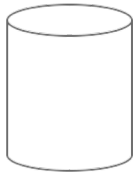


Region Proposals

Crop + Warp




Forward pass



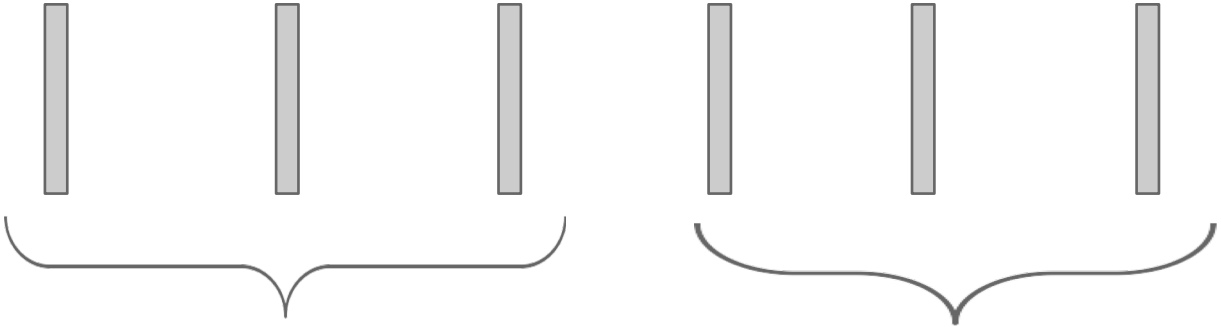
Save to disk

# 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

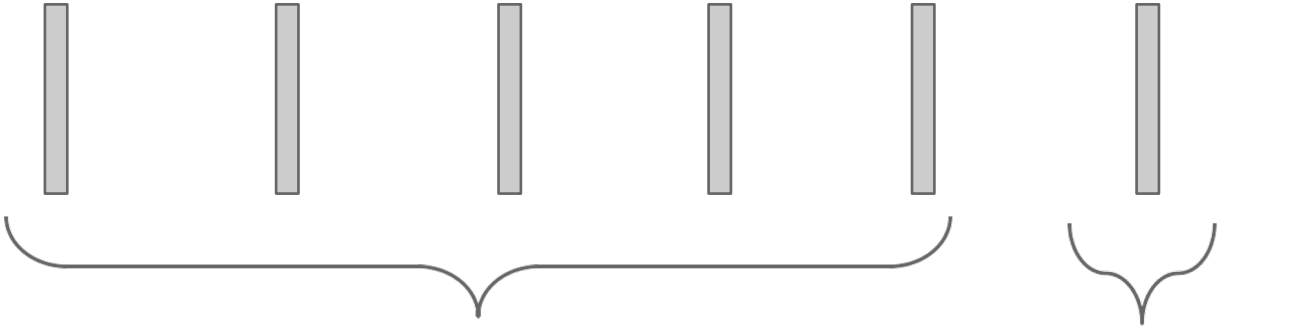
Negative samples for cat SVM

# 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

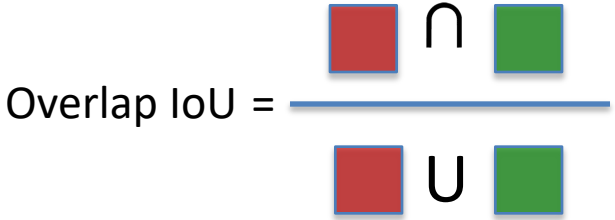
(0, 0, -0.125, 0)
Proposal too wide

# Object Detection: Datasets

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

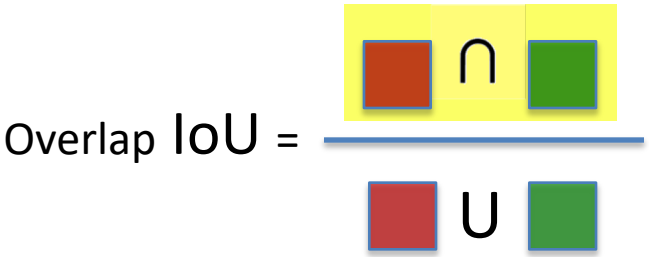
# Background: Evaluating object detectors

- Algorithm outputs ranked list of boxes with category labels
- Compute overlap between detection and ground truth box



# Background: Evaluating object detectors

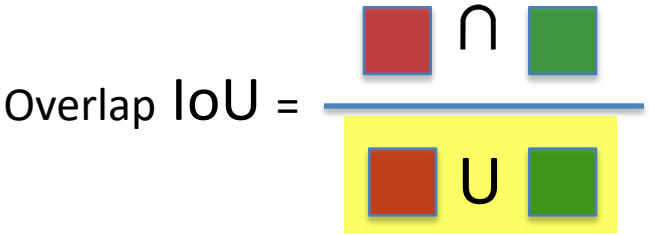
- Algorithm outputs ranked list of boxes with category labels
- Compute overlap between detection and ground truth box



# Background: Evaluating object detectors

- Algorithm outputs ranked list of boxes with category labels
- Compute overlap between detection and ground truth box





# Object Detection: Evaluation

Evaluation metric: mean average precision (mAP)

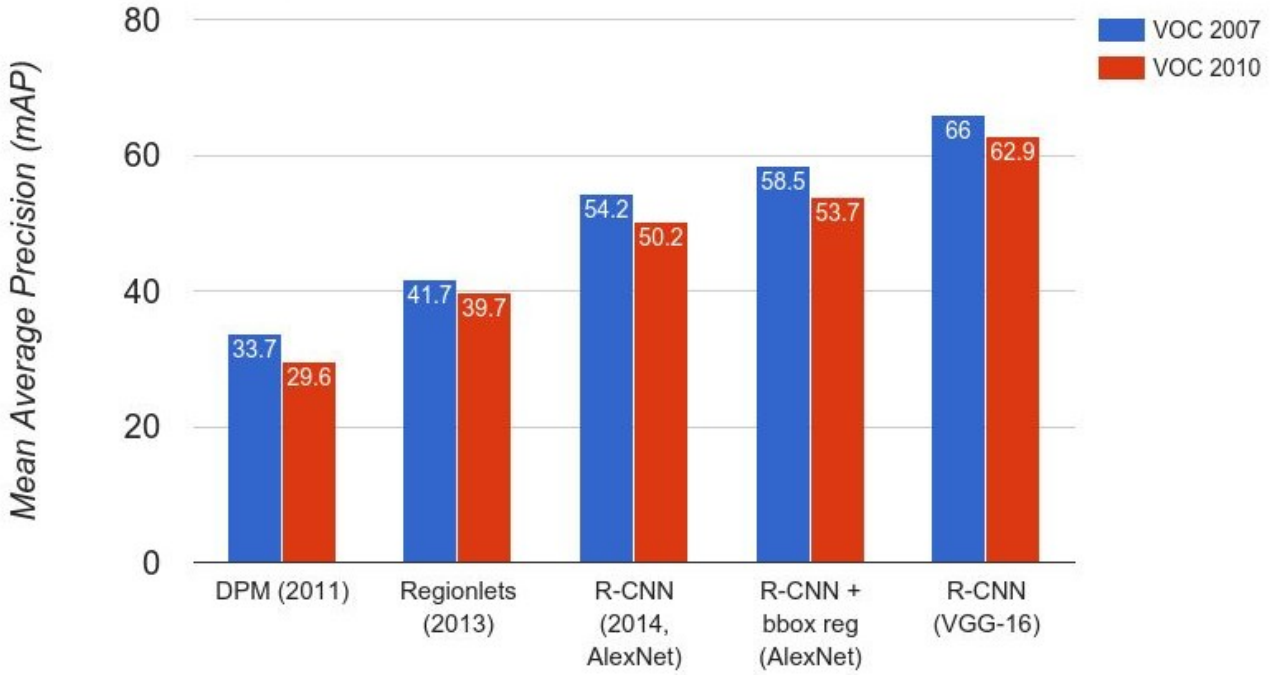
Compute average precision (AP) separately for each class, then average over classes

**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

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

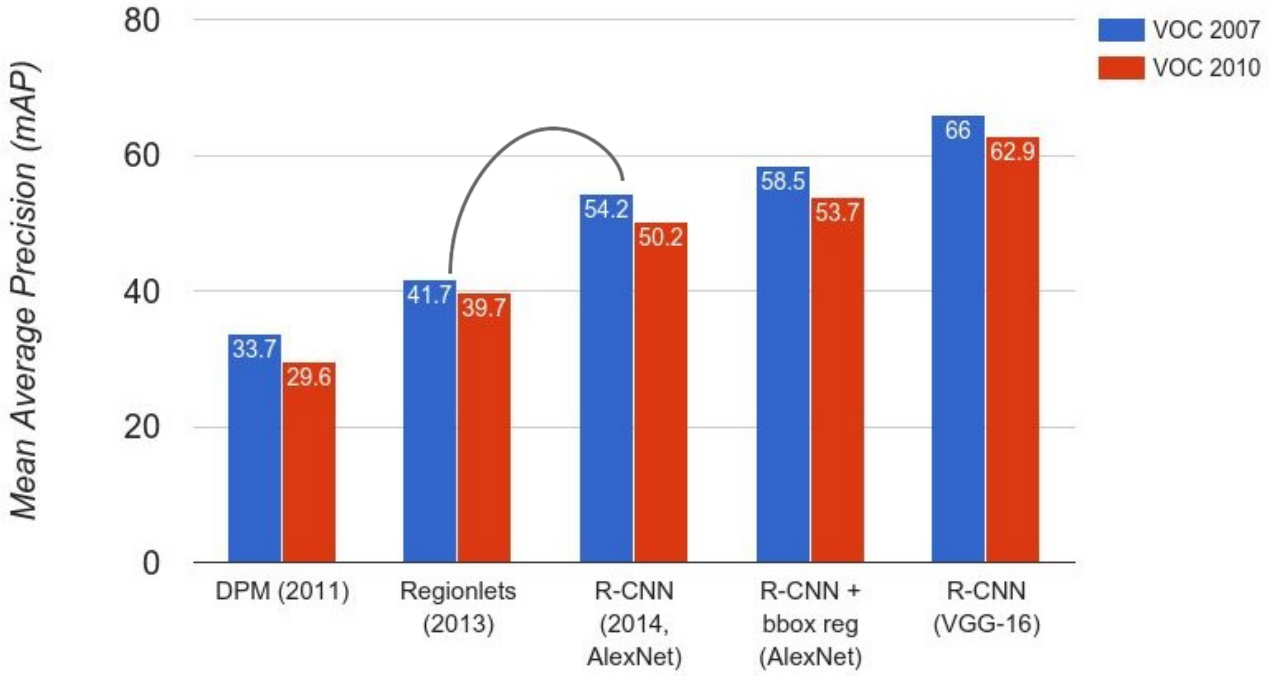
# R-CNN Results




Wang et al, “Regionlets for Generic Object Detection”, ICCV 2013

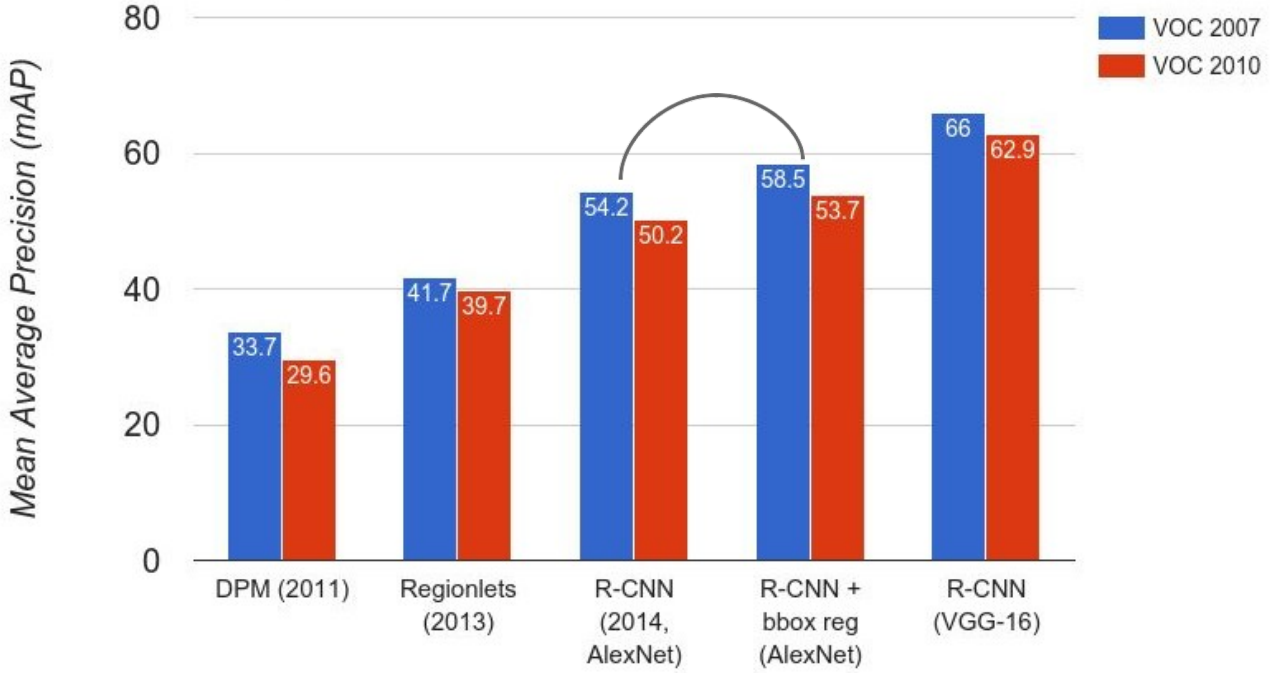
# R-CNN Results

Big improvement compared to pre-CNN methods



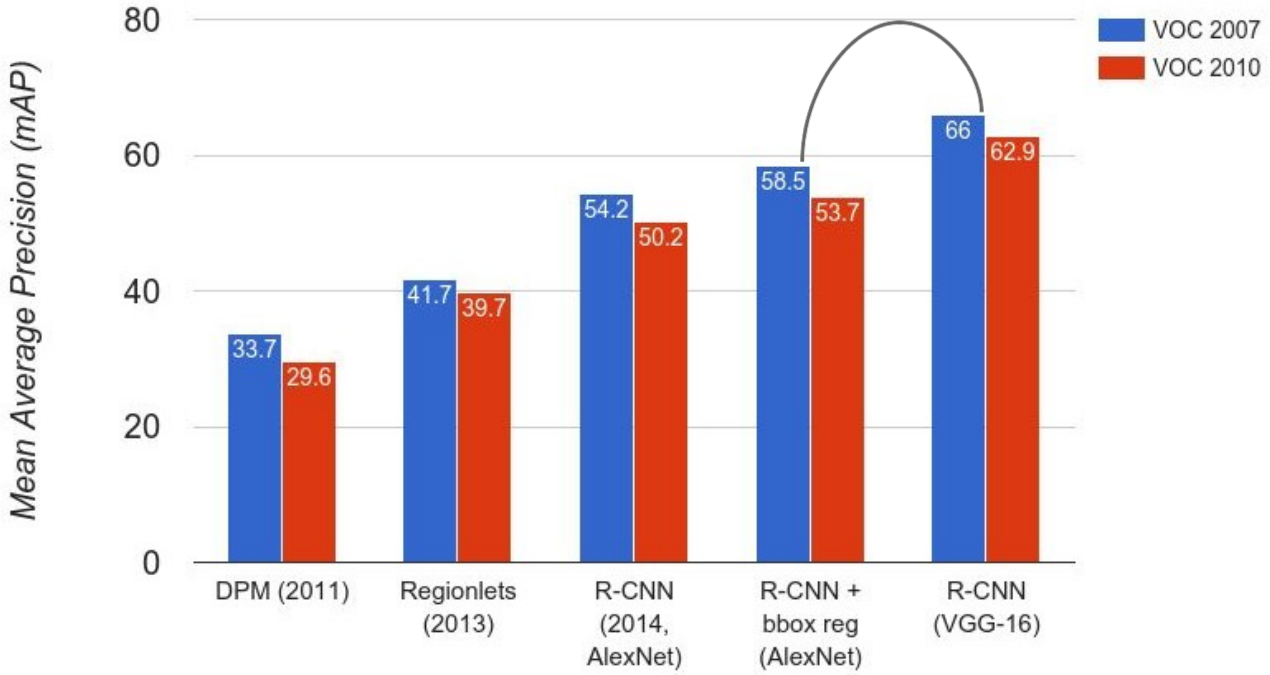
# R-CNN Results

Bounding box regression helps a bit



# R-CNN Results

Features from a deeper network help a lot

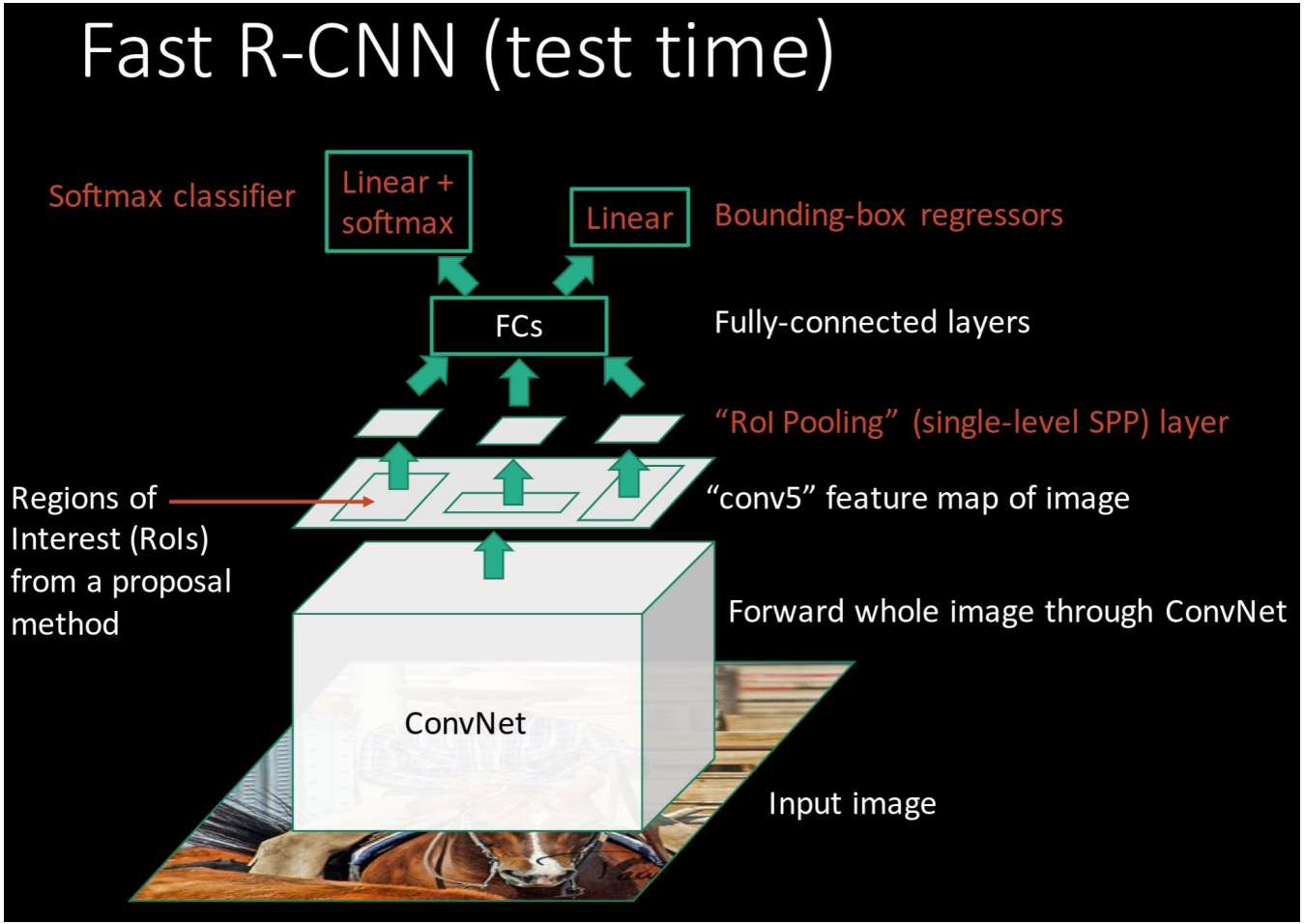


# R-CNN Problems

1. Slow at test-time: need to run full forward pass of CNN for each region proposal

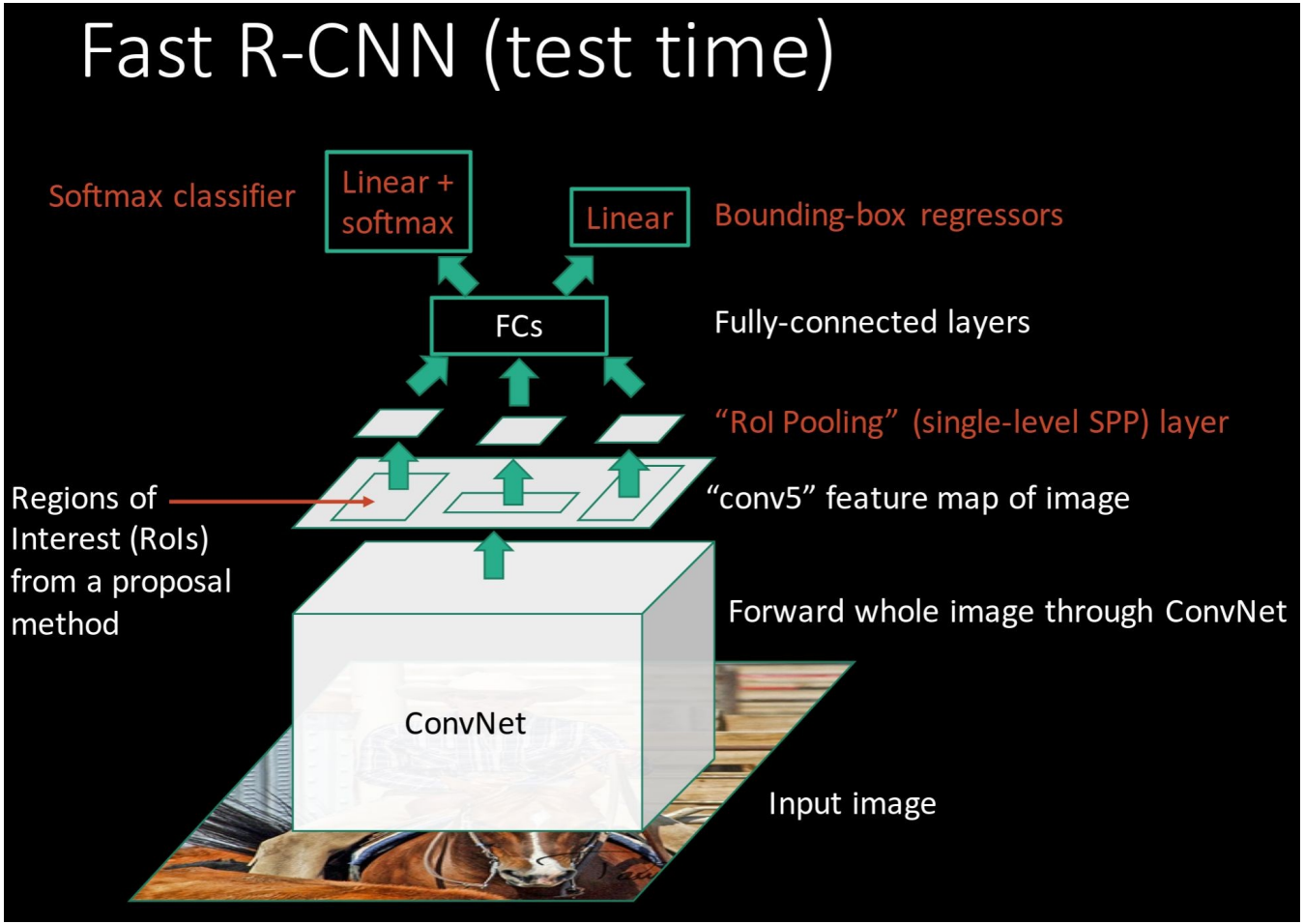
2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors

3. Complex multistage training pipeline



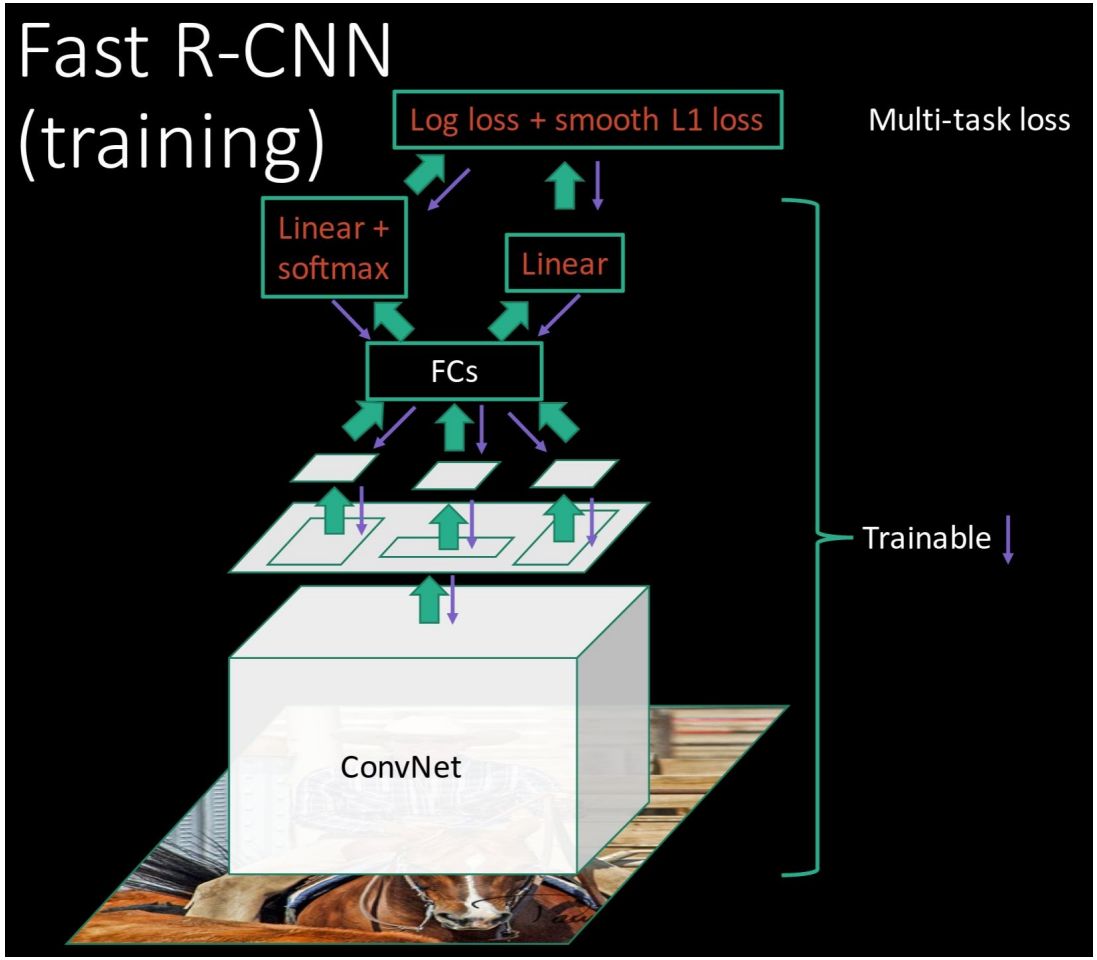

Girschick, "Fast R-CNN", ICCV 2015

Slide credit: Ross Girschick




**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




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

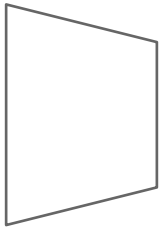
**R-CNN Problem #3:**
Complex training pipeline

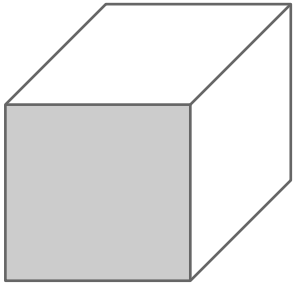
**Solution:**
Just train the whole system end-to-end all at once!

Slide credit: Ross Girschick

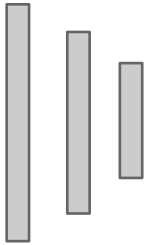
# Fast R-CNN: Region of Interest Pooling

Convolution and Pooling





Fully-connected layers

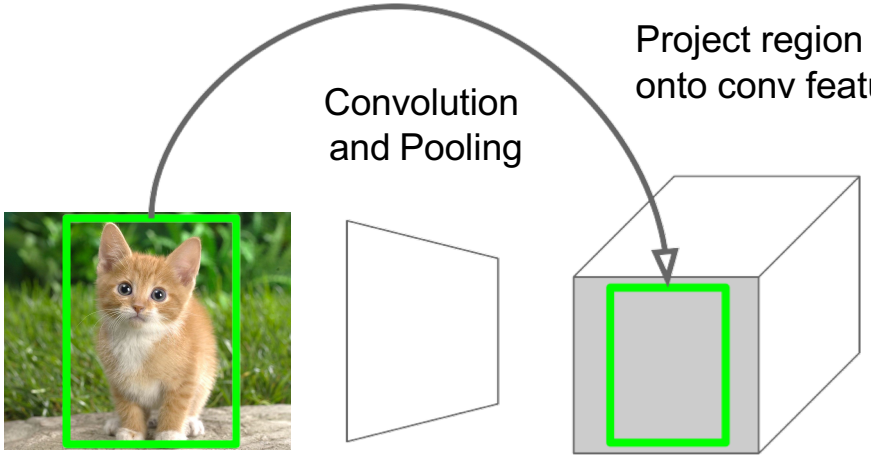


Hi-res input image:
3 x 800 x 600
with region proposal

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

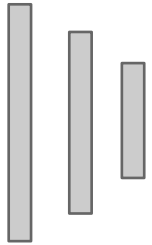
**Problem**: Fully-connected layers expect low-res conv features: C x h x w

# Fast R-CNN: Region of Interest Pooling




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

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

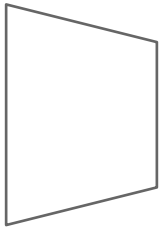


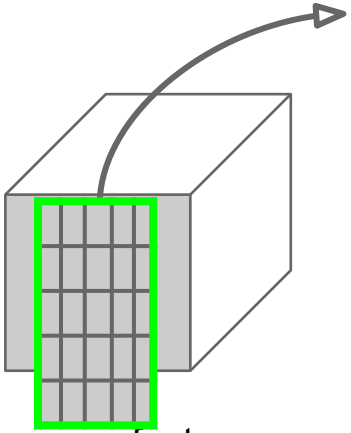

**Problem**: Fully-connected layers expect low-res conv features: C x h x w

# Fast R-CNN: Region of Interest Pooling

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

Convolution
and Pooling

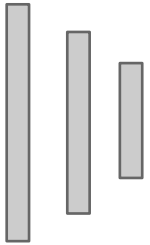




Hi-res conv features:
C x H x W
with region proposal

Divide projected
region into h x w grid

Fully-connected
layers

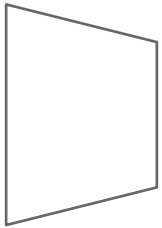


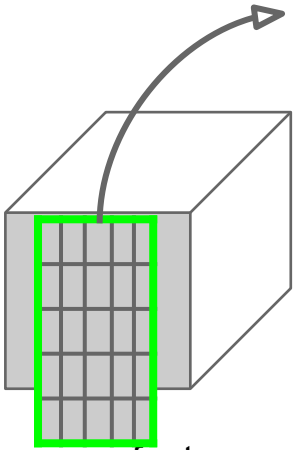
**Problem**: Fully-connected
layers expect low-res conv
features: C x h x w

# Fast R-CNN: Region of Interest Pooling

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

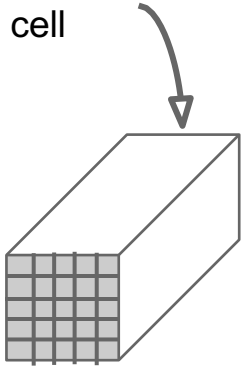
Convolution
and Pooling





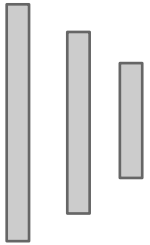
Hi-res conv features:
C x H x W
with region proposal

Max-pool within
each grid cell



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

Fully-connected
layers

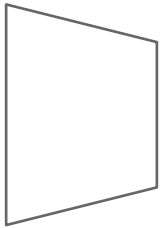


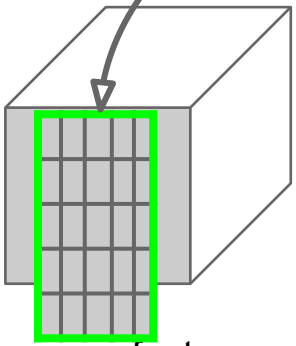
Fully-connected layers expect
low-res conv features:
C x h x w

# Fast R-CNN: Region of Interest Pooling

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

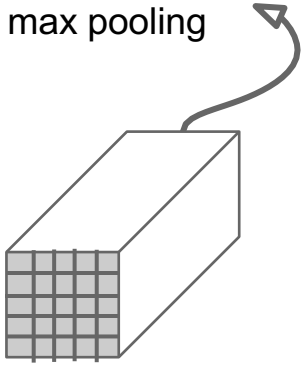
Convolution
and Pooling





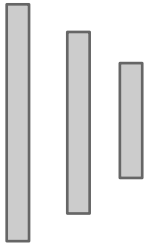
Hi-res conv features:
C x H x W
with region proposal

Can back propagate
similar to max pooling



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

Fully-connected
layers



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

# Fast R-CNN Results

| | 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** |

Faster!

FASTER!

Better!

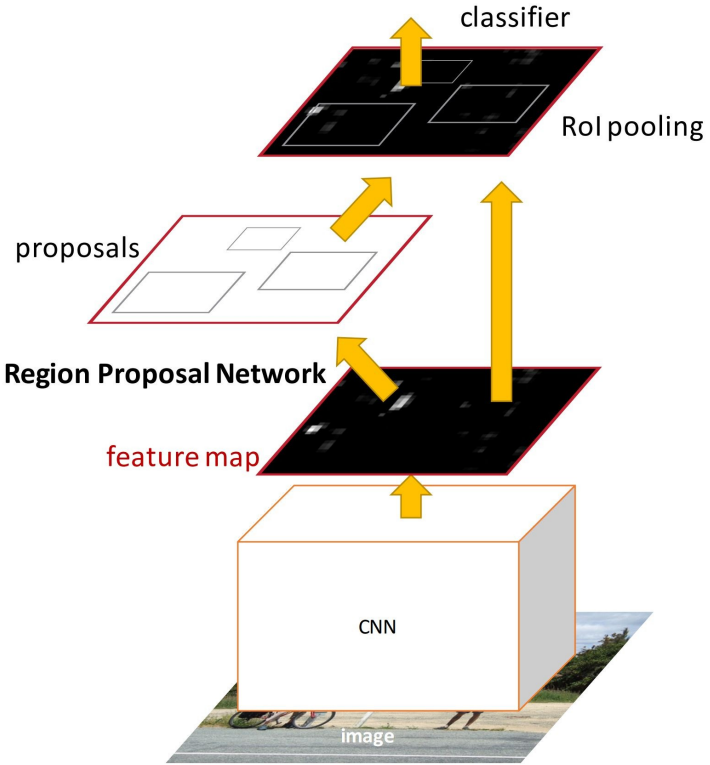
Using VGG-16 CNN on Pascal VOC 2007 dataset

# 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 Girschick

# Faster R-CNN: Training

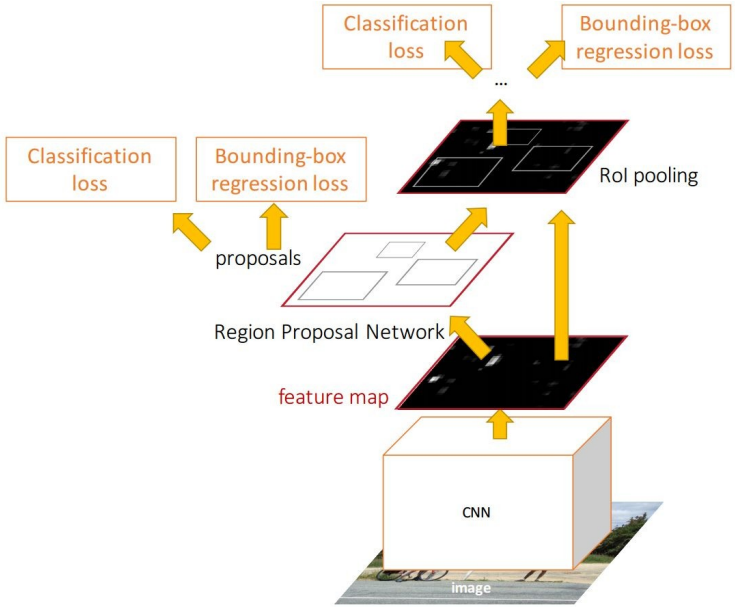
In the paper: Ugly pipeline

- 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 -> proposal)
- Fast R-CNN classification (over classes)
- Fast R-CNN regression (proposal -> box)




Slide credit: Ross Girschick

# 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** |

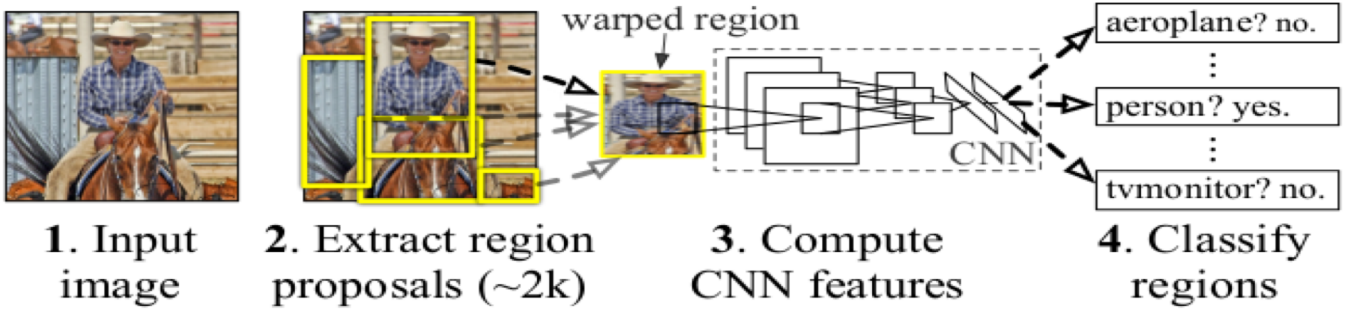
# Object Detection State-of-the-art: ResNet 101 + Faster R-CNN + some extras

| 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** |

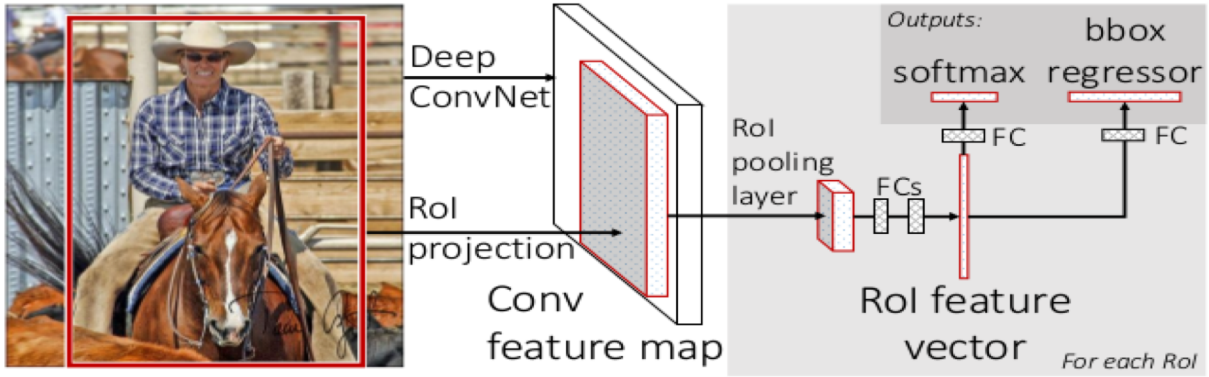
He et. al, “Deep Residual Learning for Image Recognition”, arXiv 2015

# To sum up: Detection

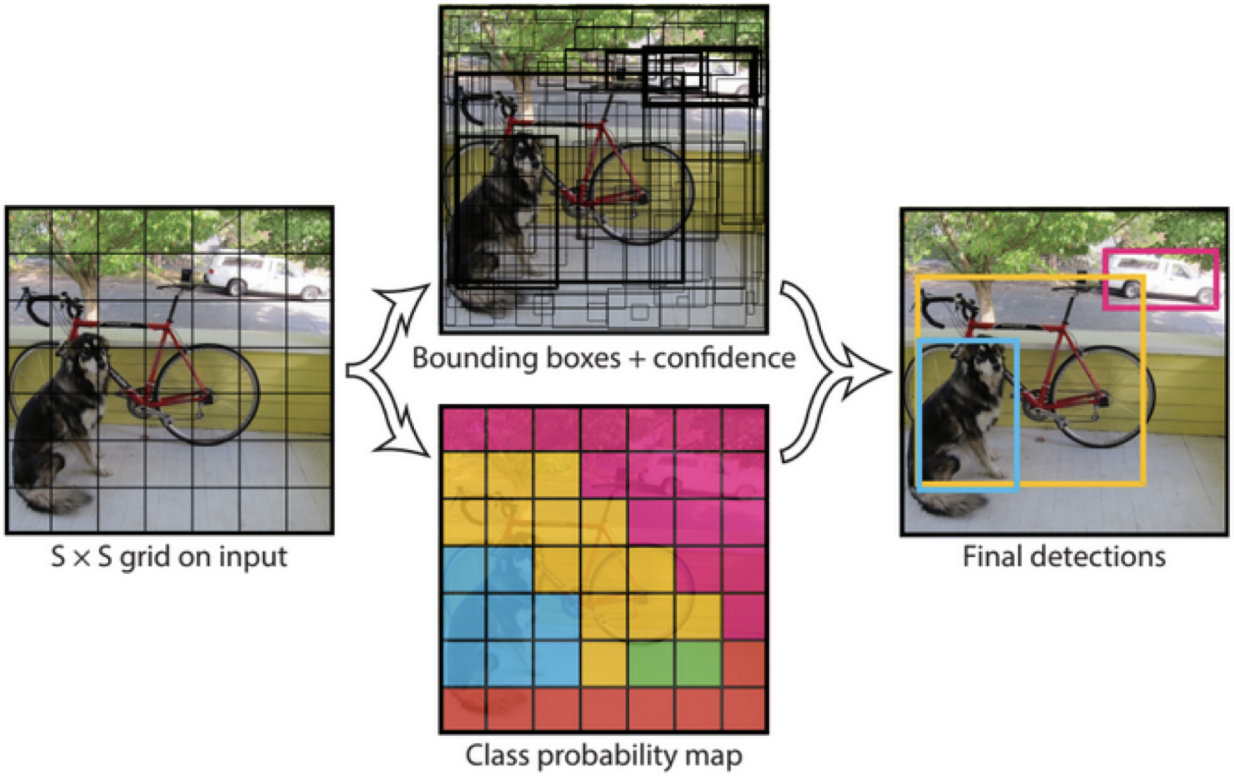
## Region CNN (R-CNN) approach




## Fast(er) R-CNN approach



Other approaches: Yolo (You Only Look Once ), SSD (Single Shot Detector) , RetinaNet, …



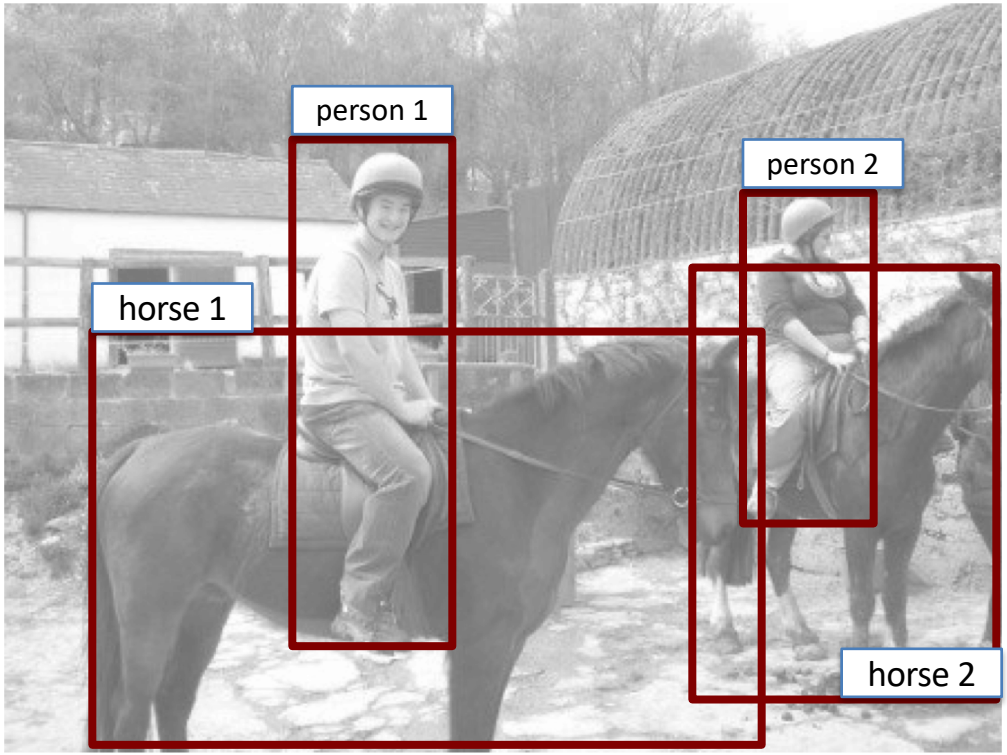
# Outline

1. Neural Nets
2. Deep Convolutional Neural Networks
3. Modern Deep Architectures
4. Beyond ImageNet
    1. Fully Convolutional Networks (FCNs)
    2. Transfer
    3. Detection
    4. **Segmentation**

# Semantic segmentation: One step further for visual scene understanding

# Object Detection

*Detect every instance of the category and localize it with a bounding box.*



# Semantic Segmentation

*Label each pixel with a category label*

# Simultaneous Detection and Segmentation

*Detect and segment every instance of the category in the image*

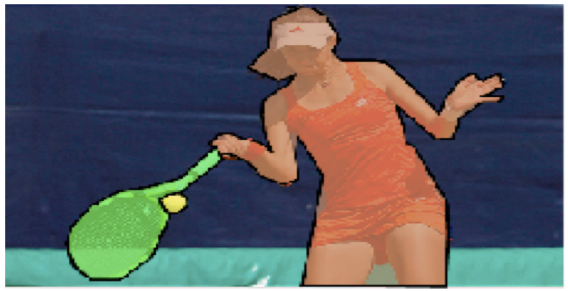



# Simultaneous Detection, Segmentation and Part Labeling

*Detect and segment every instance of the category in the image and label its parts*




# **Supervised** Segmentation with Deep ConvNets

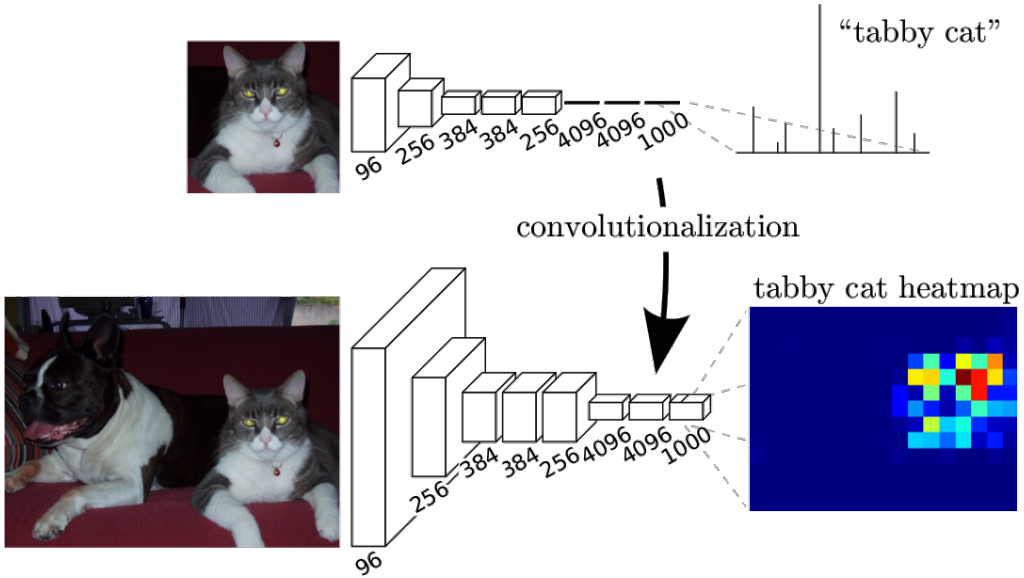


# Recent Deep strategies for Supervised Segmentation

1. **F-CN Fully Convolutional Network**
2. DeepLab approach for supervised segmentation
3. Deconvolution Networks

# FCN: Fully-Convolutional Network

- CNN: state-of-the-art model for image classification

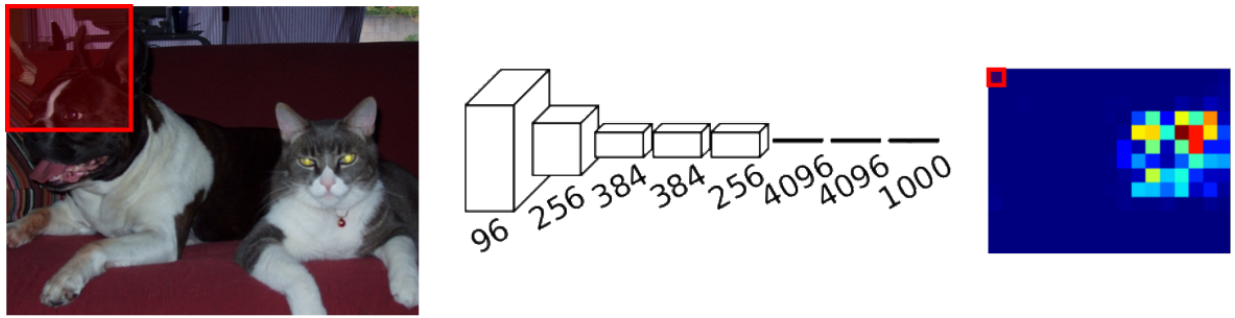



- Fully-convolutional network: classify each "pixel"

Jonathan Long, Evan Shelhamer, and Trevor Darrell.
Fully convolutional networks for semantic segmentation.
In *CVPR*, 2015. [CVPR][PAMI]

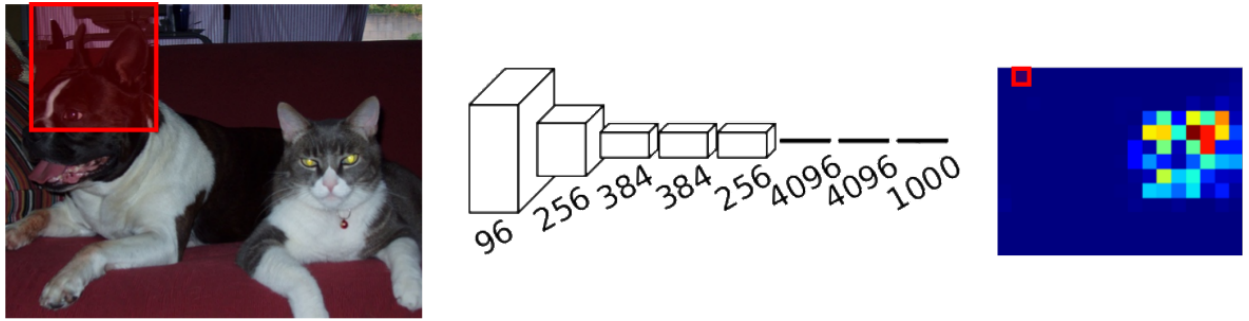
# FCN: Fully-Convolutional Network

- Classify each region



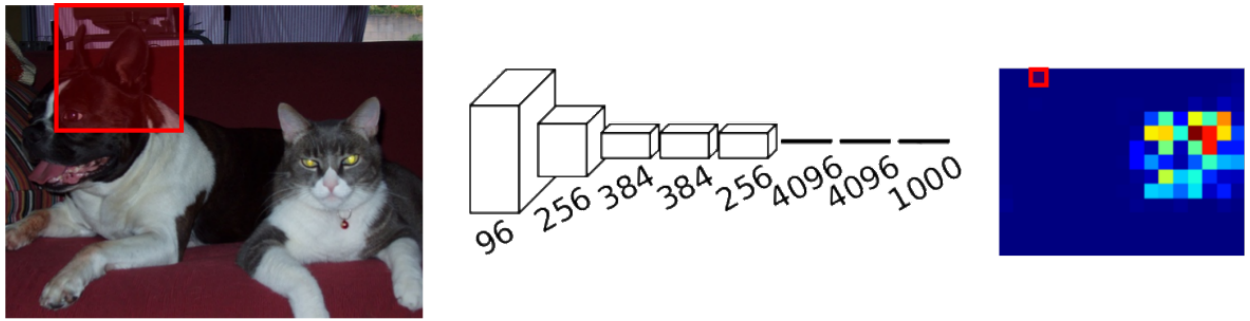
# FCN: Fully-Convolutional Network

- Classify each region



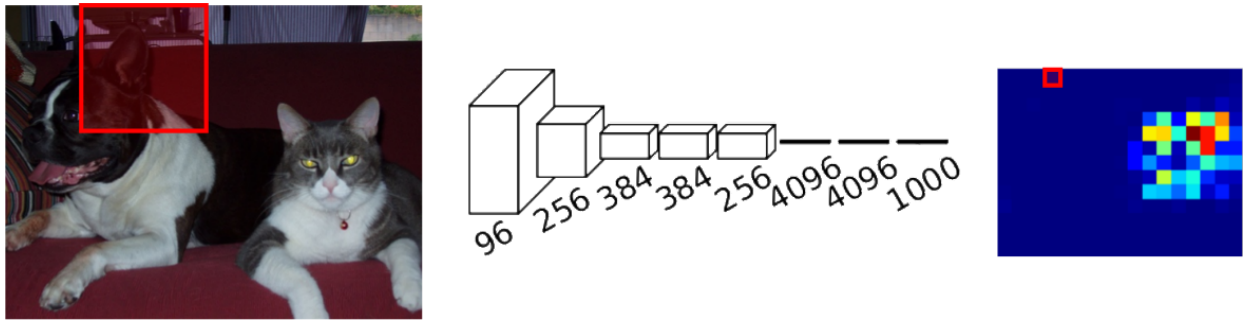
# FCN: Fully-Convolutional Network

- Classify each region



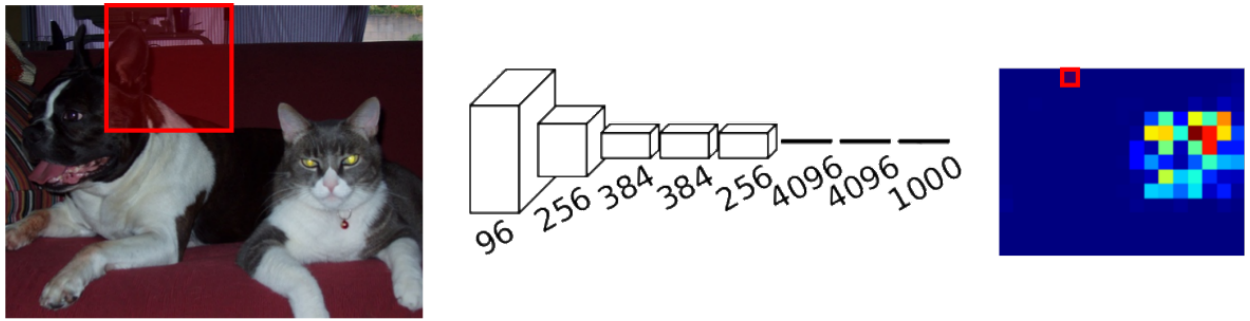
# FCN: Fully-Convolutional Network

- Classify each region



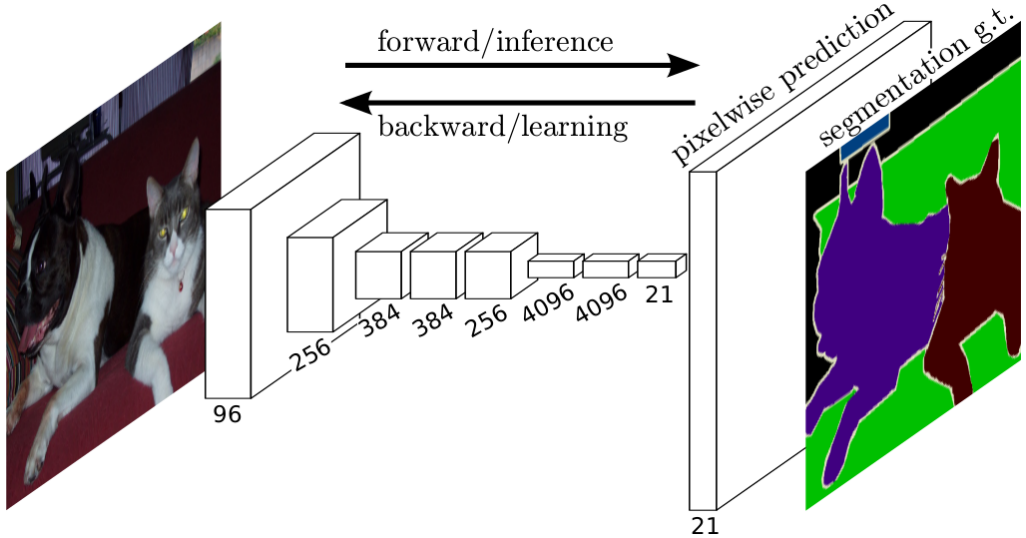
# FCN: Fully-Convolutional Network

- Classify each region



# FCN: Fully-Convolutional Network

- Fully-convolutional network: classify each "pixel"
- Upsampling output (bilinear interpolation + deconvolution)
- Network architecture: AlexNet, VGG16, GoogleNet
- Loss: soft-max per pixel



# FCN: Fully-Convolutional Network

Learning process

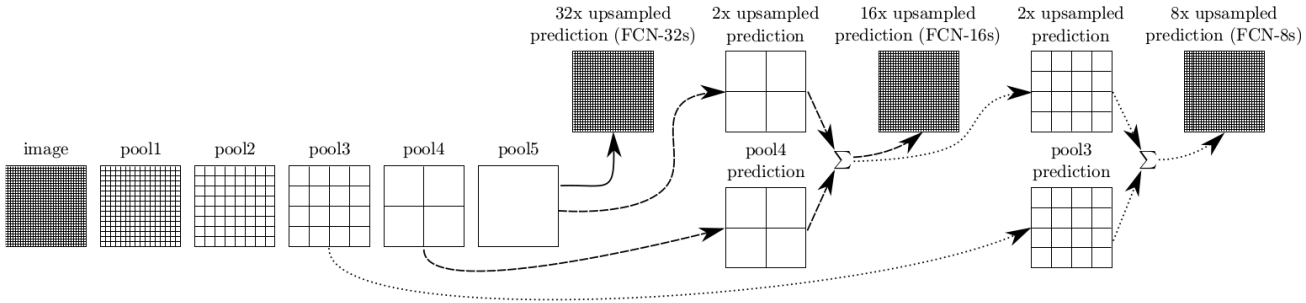
1. Model pretrained on ImageNet
2. Decapitate each net by discarding the final classifier layer
3. Convert all fully-connected layers to convolutions
4. Append $n$[1] $1 \times 1$ convolutions
5. Fine-tuning all layers by backpropagation

[1] n=number of classes

# FCN: Fully-Convolutional Network

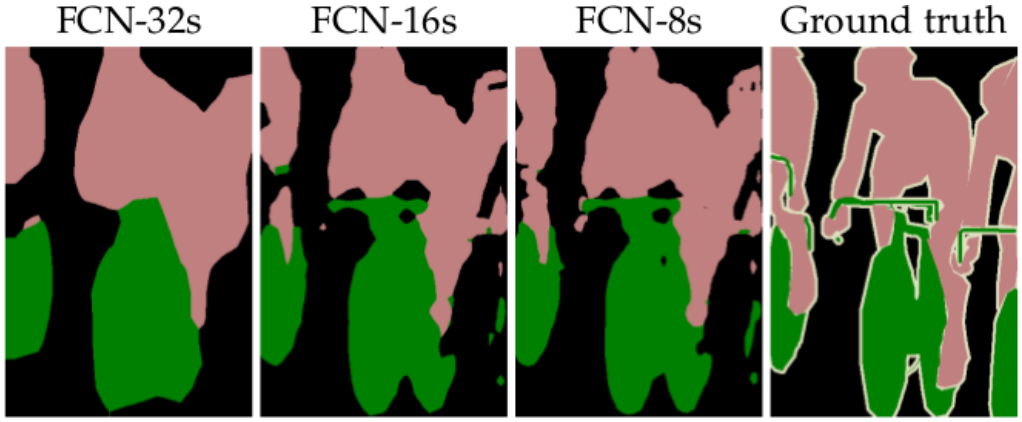
Solution of the FCN approach

- Problem: max pooling and striding reduces spatial resolution
- Dense prediction: combines feature hierarchies
- Initialized with the parameters of coarse net
- Fine-tuning all layers by backpropagation



# FCN: Fully-Convolutional Network

| | pixel acc. | mean acc. | mean IU | f.w. IU |
|---|---|---|---|---|
| FCN-32s-fixed | 83.0 | 59.7 | 45.4 | 72.0 |
| FCN-32s | 89.1 | 73.3 | 59.4 | 81.4 |
| FCN-16s | 90.0 | 75.7 | 62.4 | 83.0 |
| FCN-8s | **90.3** | **75.9** | **62.7** | **83.2** |




FCN-32s-fixed: only the last layer is fine-tuned

# Recent Deep strategies for Supervised Segmentation

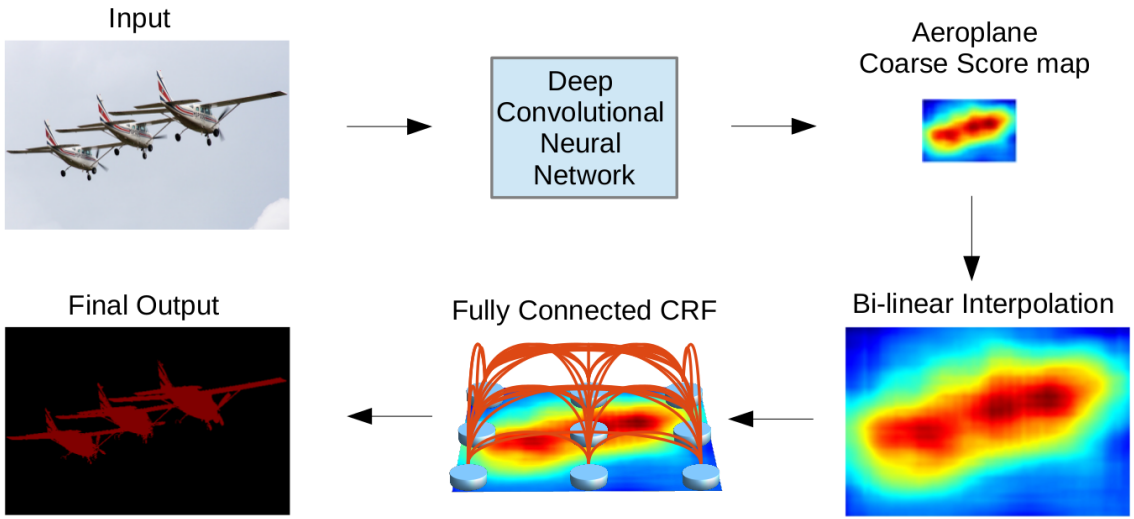
1. F-CN Fully Convolutional Network
2. **DeepLab approach for supervised segmentation**
3. Deconvolution Networks

# DeepLab [ICLR 15] approach for supervised segmentation

Problem of the spatial resolution reduction

Solution of the DeepLab approach

1. Learn CNN for dense prediction tasks (Atrous)
2. Improve the localization of object boundaries with fully-connected CRF [?] (FC-CRF)



# Atrous ('Holes') Algorithm

- Remove the down-sampling from the last pooling layers.
- Up-sample the original filter by a factor of the strides:

  Atrous convolution for 1-D signal:

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k] w[k]$$

Introduce zeros between filter values

$x[i]$ 1-D input signal

$w[k]$ filter of length $K$

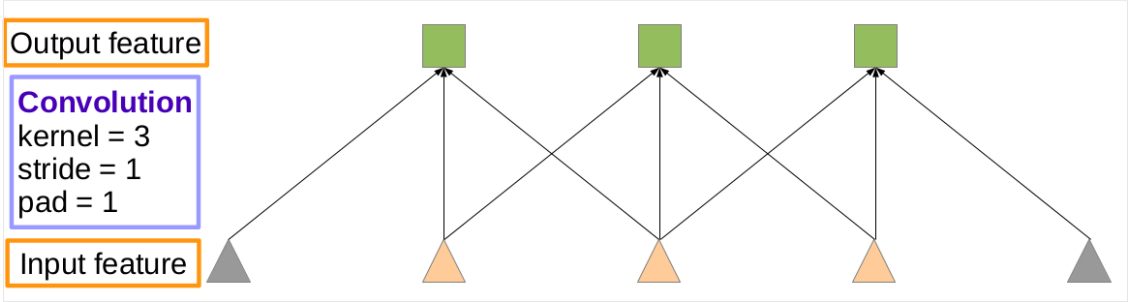
$r$ *rate* parameter corresponds to the stride with which we sample the input signal.

$y[i]$ output of atrous convolution.

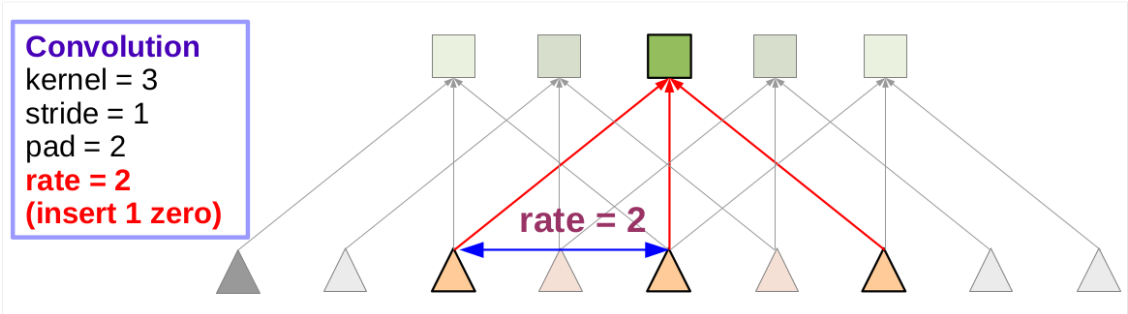
- Note: standard convolution is a special case for *rate* $r$=1.

# DeepLab: Atrous (Dilated) Convolution

- Efficient dense feature extraction
- Upsample filters




(a) Sparse feature extraction




(b) Dense feature extraction

# Atrous ('Holes') Algorithm

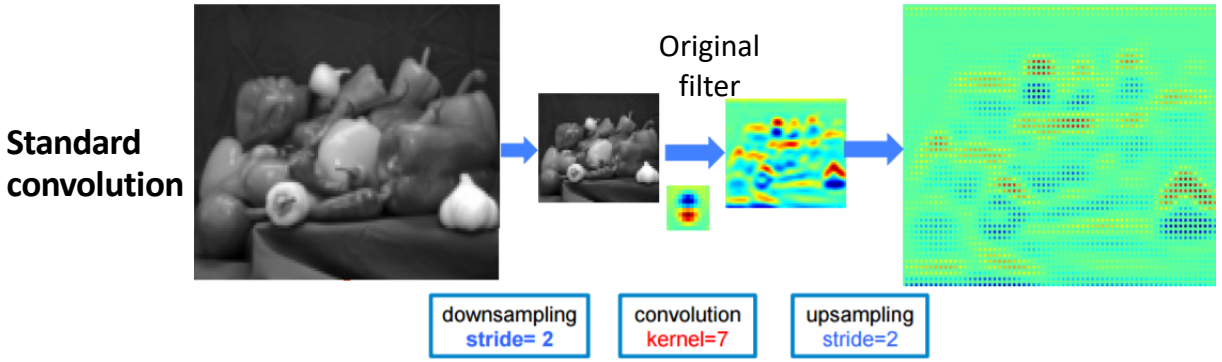
## Filters field-of-view

- **Small** field-of-view → accurate **localization**
- **Large** field-of-view → **context** assimilation
- 'Holes': Introduce zeros between filter values.
- **Effective filter size increases** (enlarge the **field-of-view** of filter):

$$k \times k \text{ filter to } k_e = k + (k-1)(r-1)$$

- However, we take into account **only** the **non-zero** filter values:
  - ✓ Number of filter parameters is the same.
  - ✓ Number of operations per position is the same.

# Atrous ('Holes') Algorithm

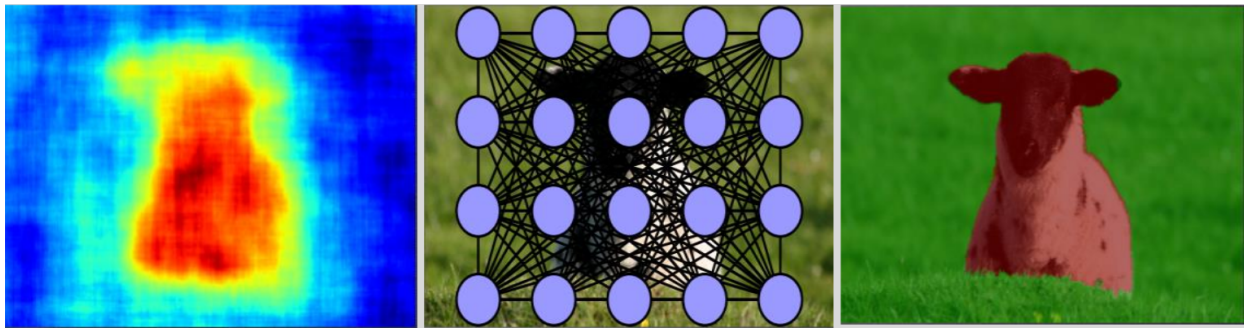


# DeepLab: Fully-Connected CRF

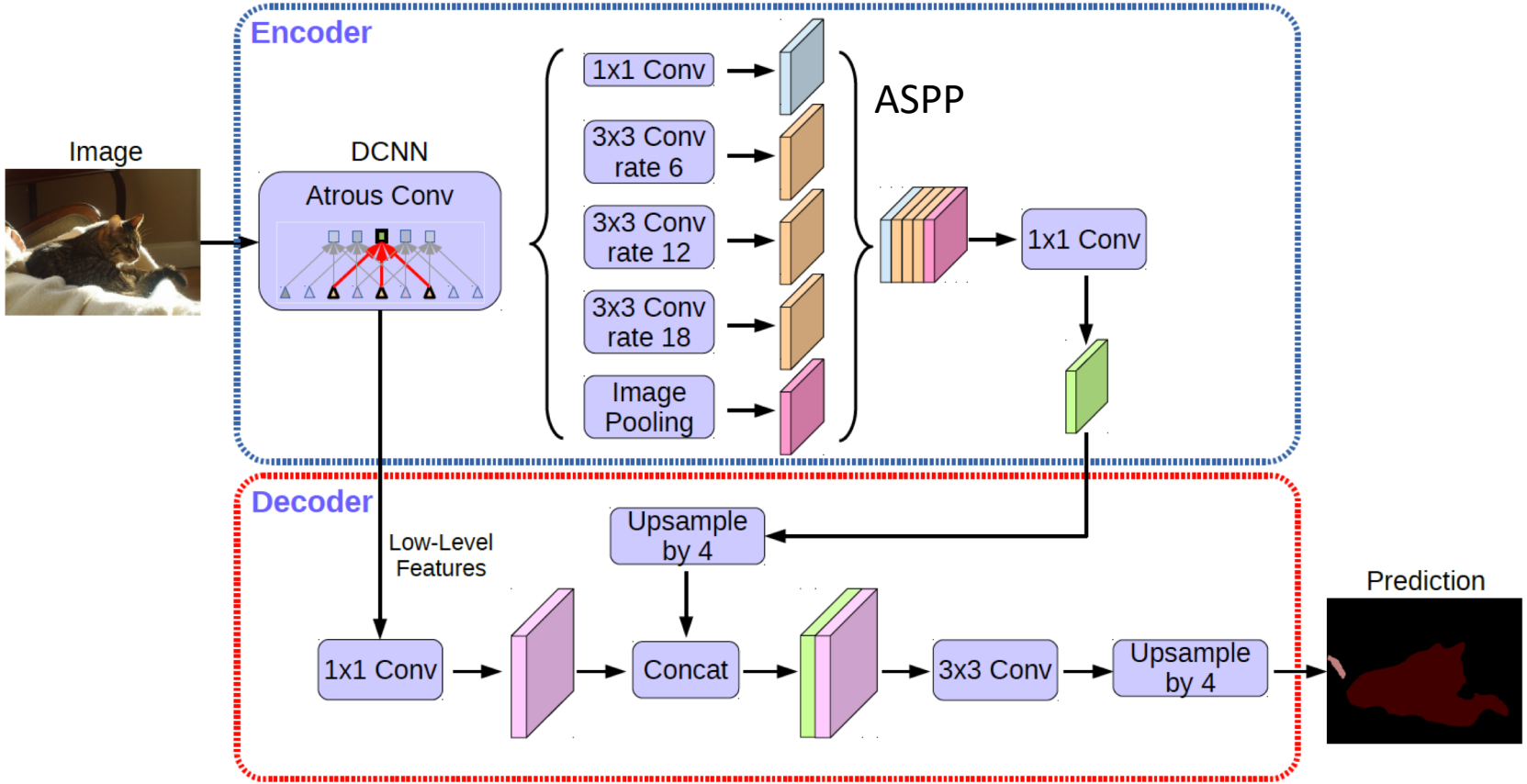
- Problem: poor object delineation (spatial and appearance consistency neglected)
- Solution: fully-connected CRF accounts for contextual information in the image

$$E(\mathbf{y}) = \sum_i \theta_i(y_i) + \sum_{i,j} \theta_{ij}(y_i, y_j)$$

  - Unary term: output of FCN (upscaled)
  - Pairwise term: penalizes similar pixels having different labels

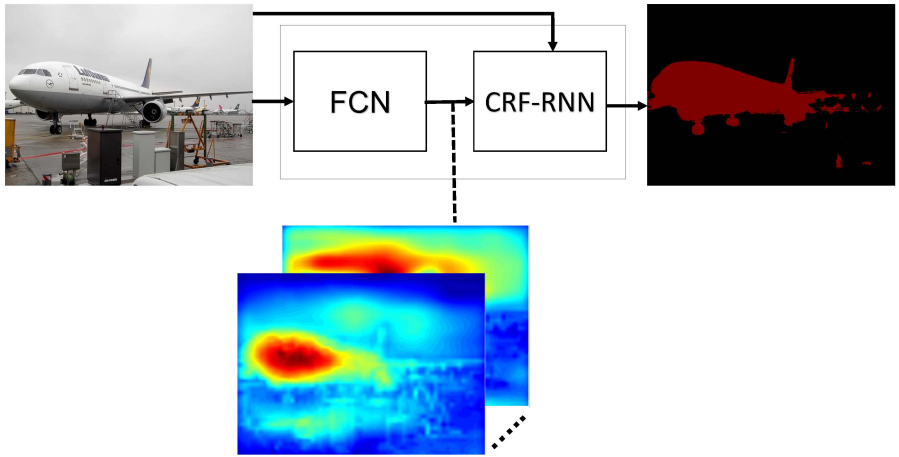


# DeepLab V3+ [ECCV 2018]



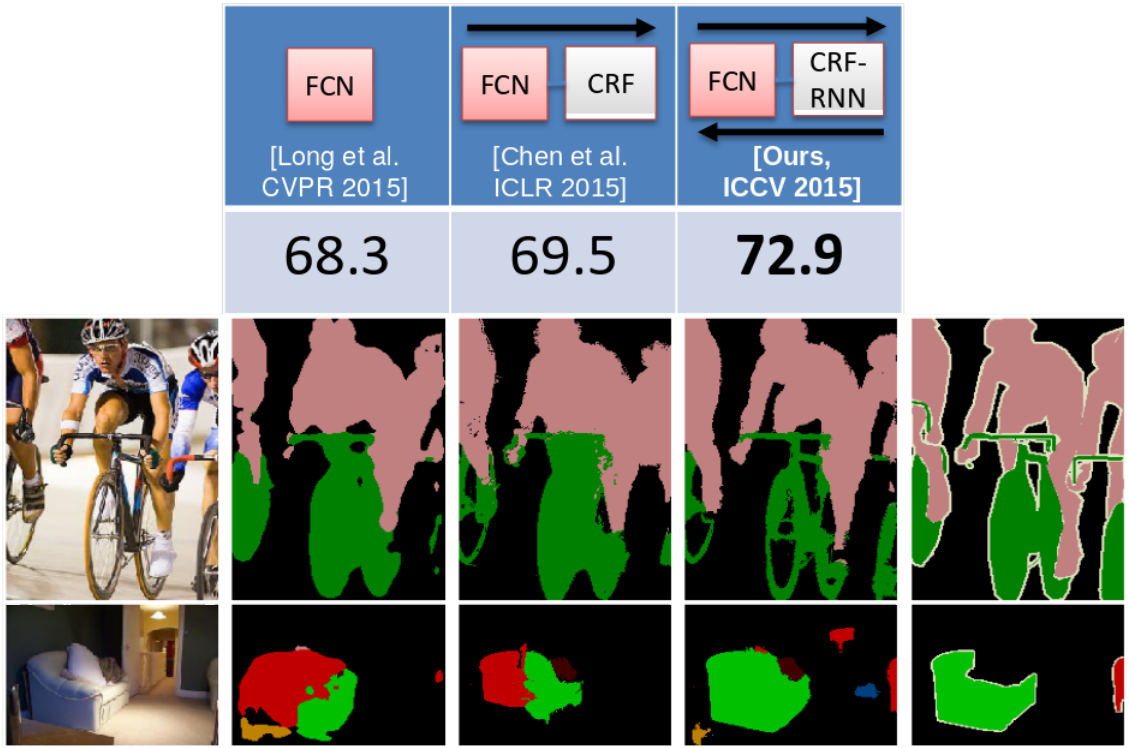
# CRF as RNN

- Unified framework: combines CNN and CRF
- Formulate mean-field approximate inference for FC-CRF as Recurrent Neural Networks (RNN)
- Learning the network end-to-end




Zheng, Jayasumana, Romera-Paredes, Vineet, Su, Du, Huang, Torr.
Conditional Random Fields as Recurrent Neural Networks.
In *ICCV*, 2015. [paper]

# CRF as RNN

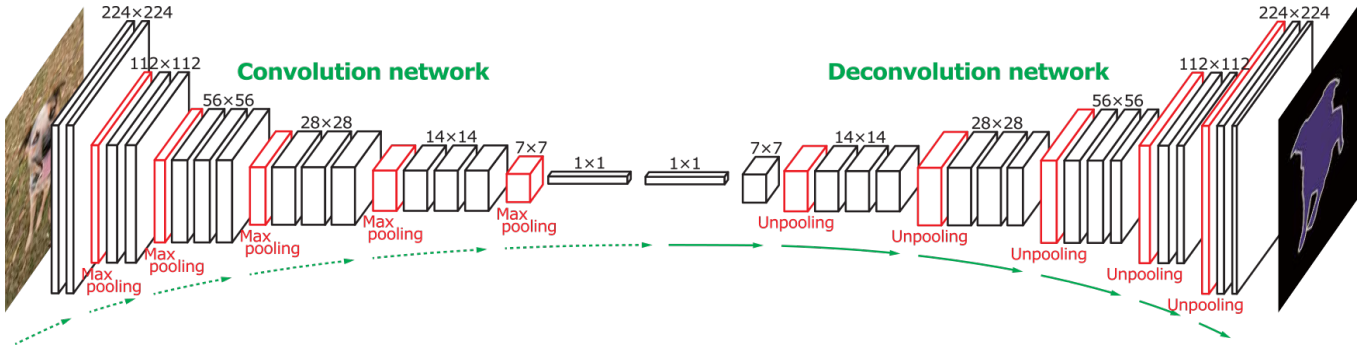

FCN
[Long et al. CVPR 2015]
FCN
CRF
[Chen et al. ICLR 2015]
FCN
CRF-RNN
[Ours, ICCV 2015]
68.3
69.5
72.9

# Recent Deep strategies for Supervised Segmentation

1. F-CN Fully Convolutional Network
2. DeepLab approach for supervised segmentation
3. **Deconvolution Networks**

# Deconvolution Network

- Learn a multi-layer deconvolution network
- Network is composed of two parts:
  1. Convolution: feature extractor
  2. Deconvolution: shape generator that produces object segmentation from the feature extracted
- Deconvolution net is a mirrored version of the convolution net



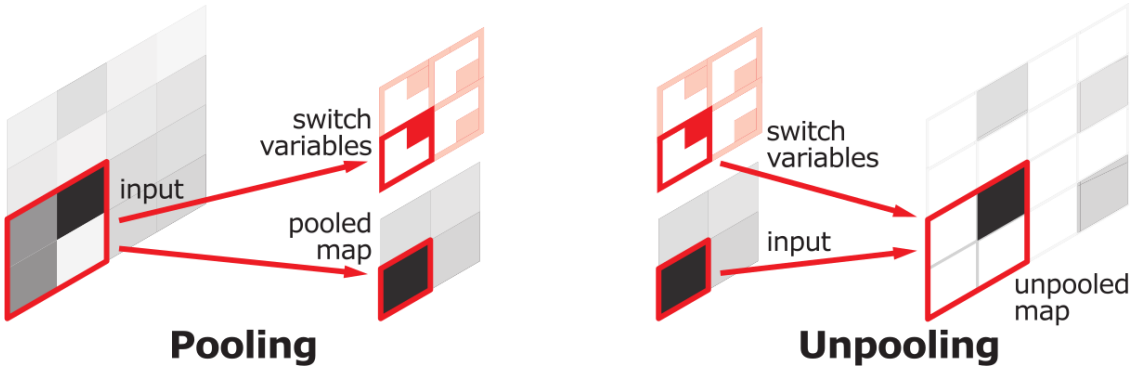


Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han.
Learning deconvolution network for semantic segmentation.
In *ICCV*, 2015. [paper]

# Deconvolution Network

## Unpooling

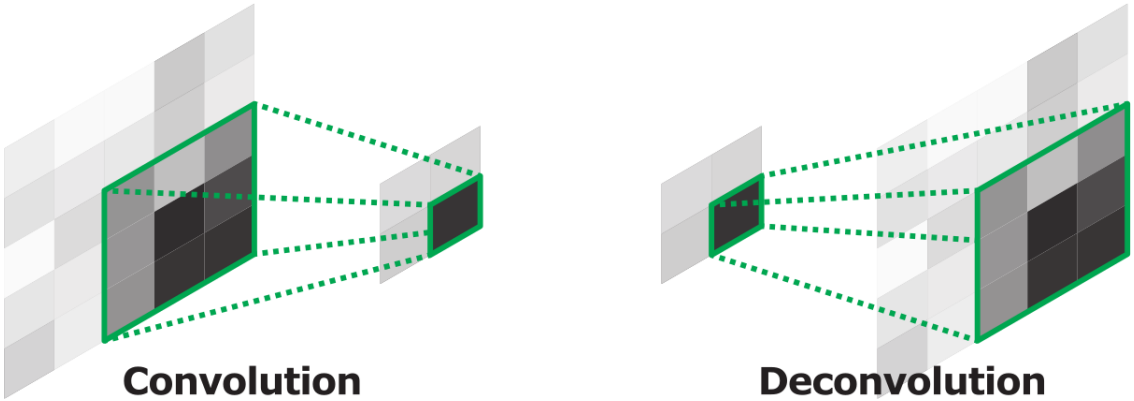
- Perform the reverse operation of pooling
- Reconstruct the original size of activations
- Useful to reconstruct the structure of input object
- Output: sparse activation map



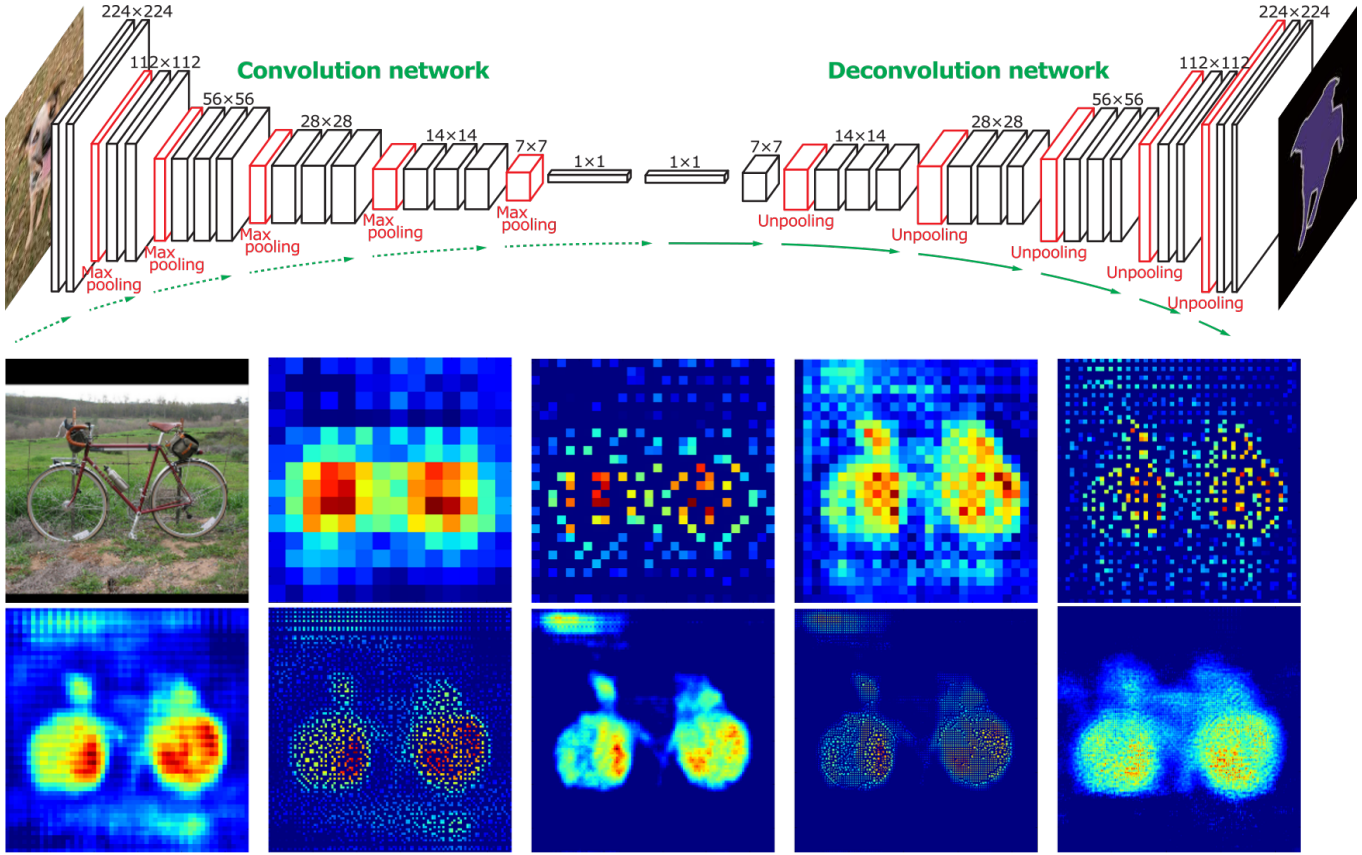
# Deconvolution Network

## Deconvolution

- Connect single input activation to a multiple activations
- Learned filters correspond to bases to reconstruct shape of an input object
- Output: enlarged and dense activation map



# Deconvolution Network



# Deconvolution Net (2)

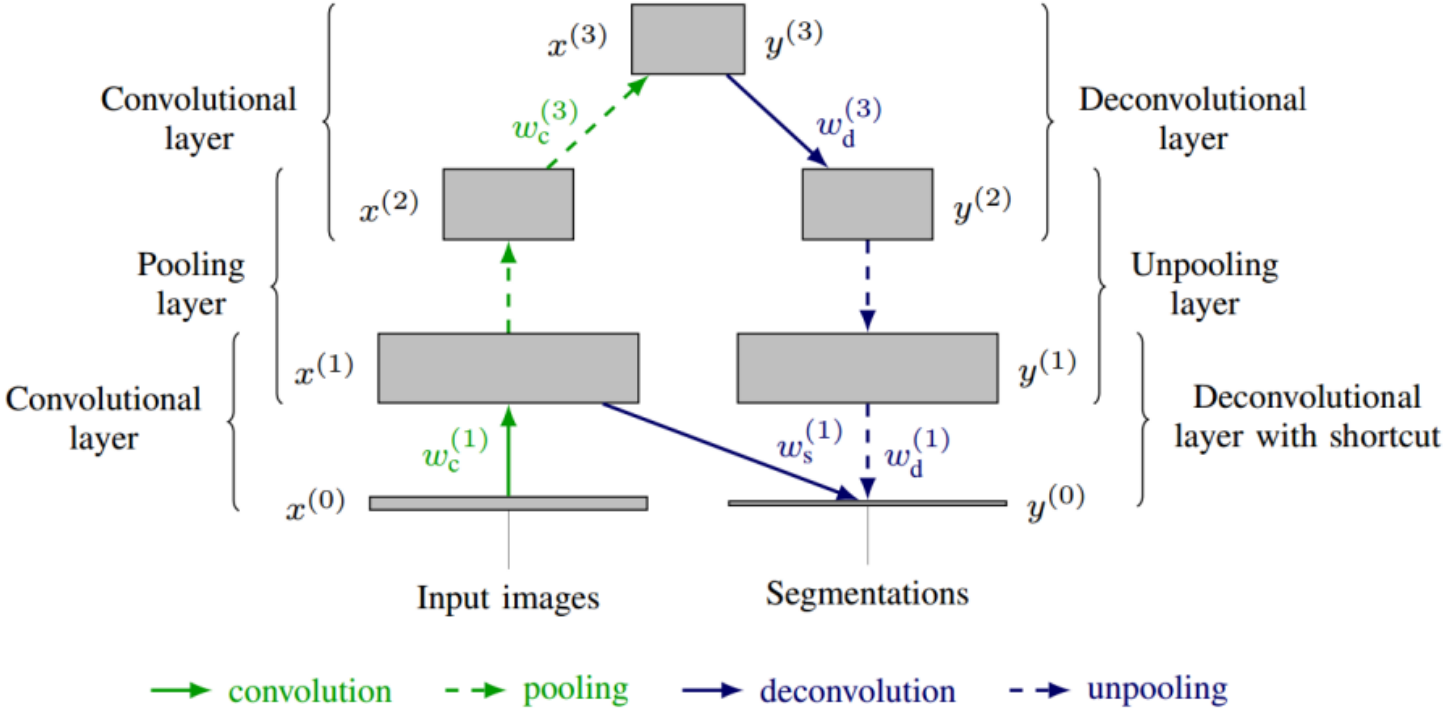



Fig. 3: CEN-s: The proposed architecture by Brosch et al. [11].

# U-Net

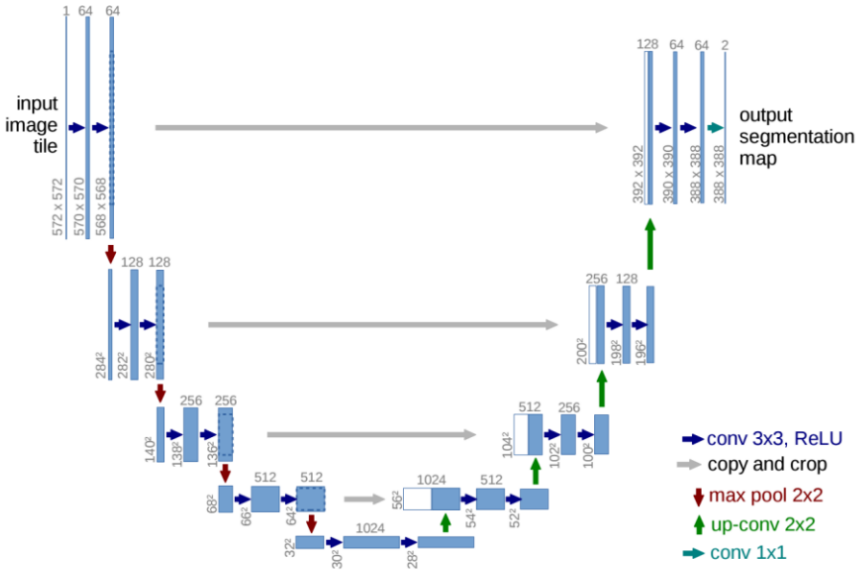



**Fig. 1.** U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Very popular in MICCAI 2016
Works well with low data

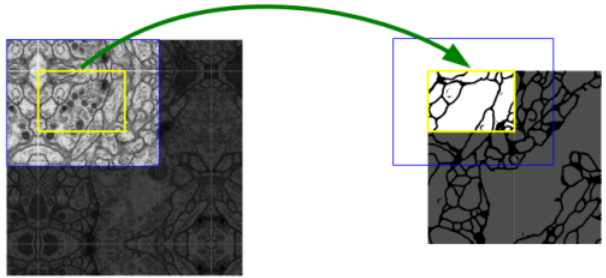


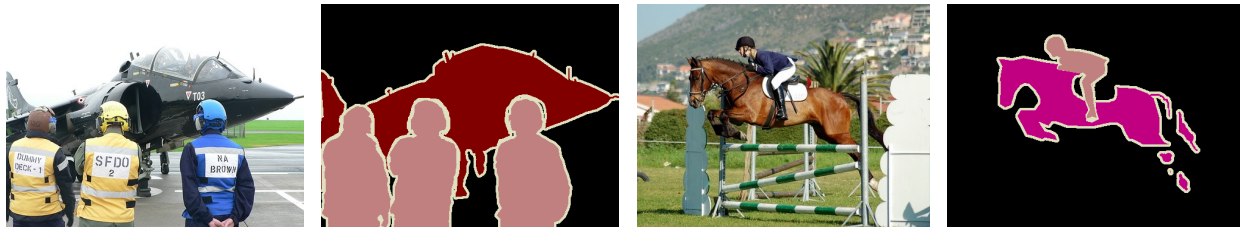
**Fig. 2.** Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring

http://deeplearning.net/tutorial/unet.html

# Datasets

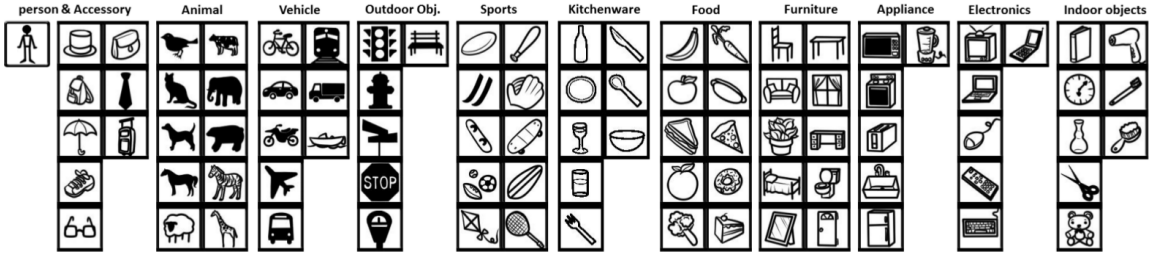
# PASCAL VOC 12

- Train 1464 images / Val 1449 images / Test 1456 images
- 21 classes: *aeroplane*, *bicycle*, *bird*, *boat*, *bottle*, *bus*, *car*, *cat*, *chair*, *cow*, *dining table*, *dog*, *horse*, *motorbike*, *person*, *potted plant*, *sheep*, *sofa*, *train*, *tv/monitor* + background
- Evaluation: intersection-over-union metric
- Webpage: `http://host.robots.ox.ac.uk/pascal/VOC/voc2012/`



# COCO

- Train 80k images / Val 20k images
- 91 classes, 11 super-categories:




- 3 challenges: detection, instance segmentation, captioning
- Webpage: `http://mscoco.org` [paper]

# Weakly Supervised Segmentation

# Supervised Image Segmentation Methods

## Full supervision

- Precise annotation ☺
- Expensive and time consuming to obtain
  - "79s per label per image" [RBFL15] ☹
- Bottleneck for learning models at large scale ☹

# Weakly Supervised Image Segmentation Methods

## Weak supervision

- Reduce supervision: class labels (or tags) ☹ ☹
- Cheap to obtain
  - "1s per label per image" [RBFL15] ☺
- Scalable to large number of categories ☺

✓ background
✗ aeroplane
✗ cat
✓ chair
✗ dog
✗ person
✗ sheep
✓ table
✗ tvmonitor

# Weakly supervised segmentation with CNN

## Standard learning algorithms

- Maximize the likelihood of the observed training data

## Problem

- Require full knowledge of the ground truth labeling
  - not available in the weakly supervised setting ☹

## Solutions

1. Generation of segmentation mask

   George Papandreou, Liang-Chieh Chen, Kevin Murphy, and Alan L. Yuille. Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. In *ICCV*, 2015.

2. Modified loss function: CNN optimized for classification

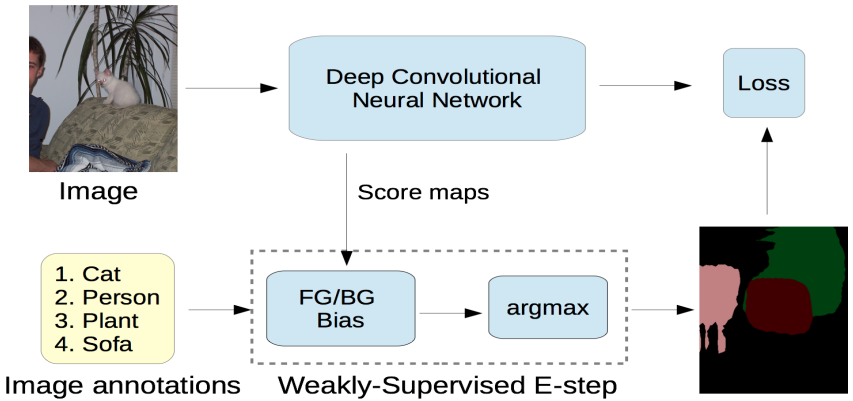
   Pedro O. Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *CVPR*, 2015.

# Generation of segmentation mask

George Papandreou, Liang-Chieh Chen, Kevin Murphy, and Alan L. Yuille.
Weakly-and semi-supervised learning of a dcnn for semantic image segmentation.
In *ICCV*, 2015.

## Idea: adaptive bias

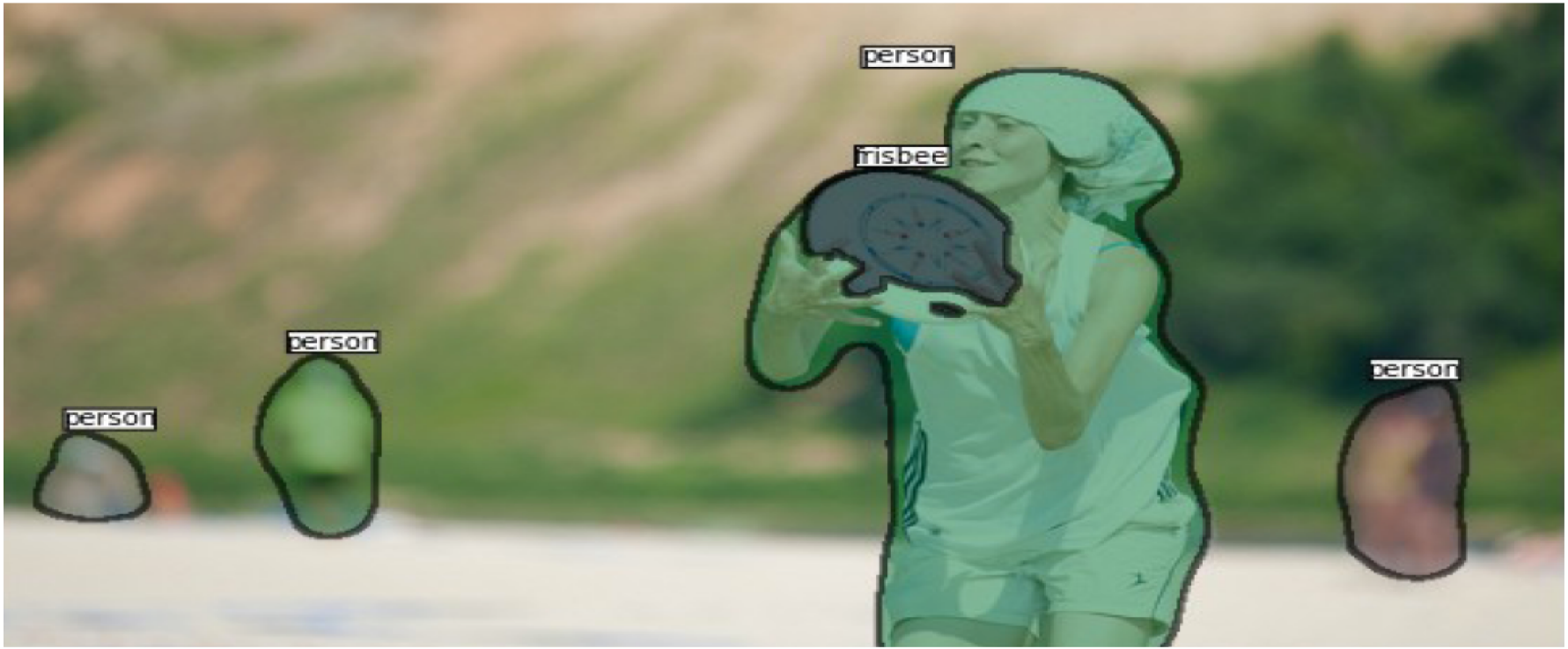
- Generated segmentation mask and train fully-supervised CNN
- Adaptive bias into the multi-instance learning framework
  - Boost classes known to be present
  - Suppress all others



# Segmentation Results

@Y. LeCun

# Segmentation Results




@Y. LeCun

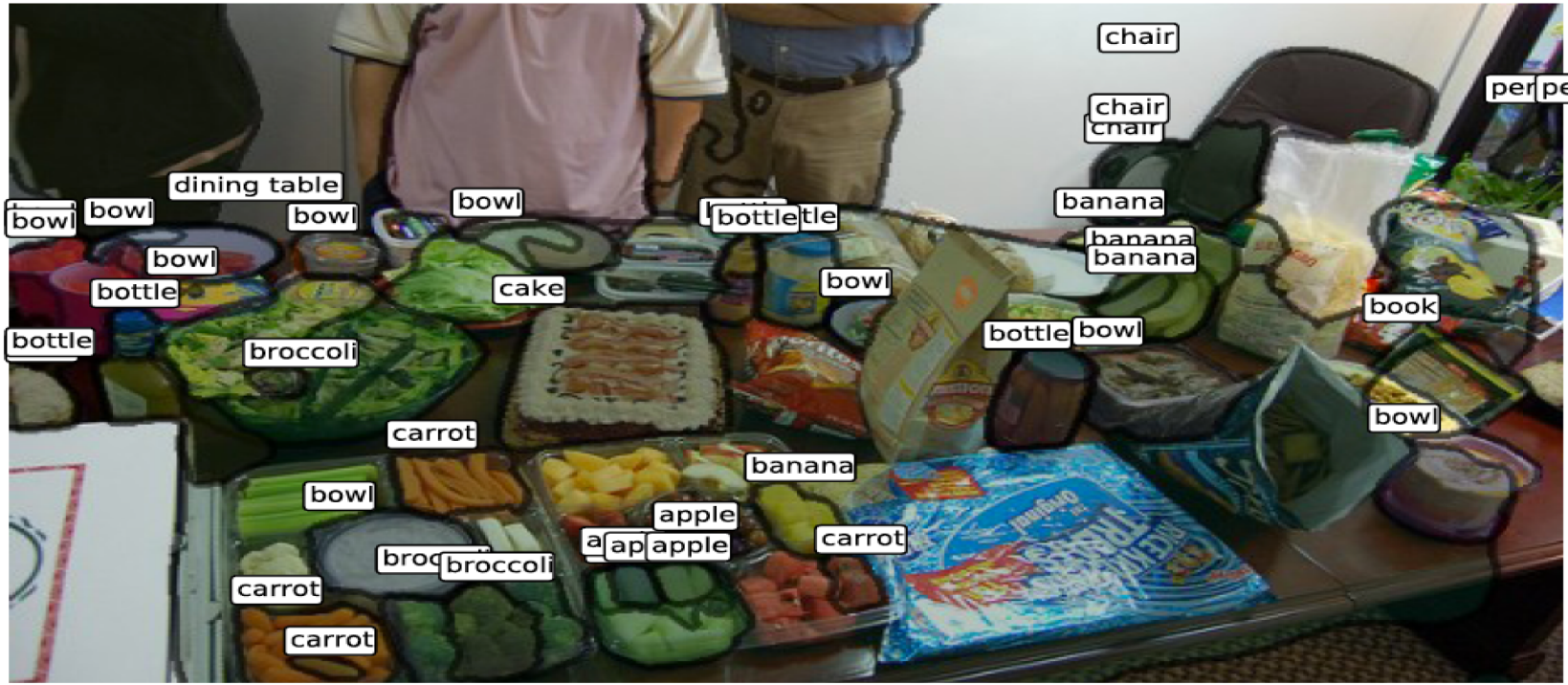
# Segmentation Results

@Y. LeCun

# Segmentation Results


@Y. LeCun

# Segmentation Results




@Y. LeCun

=> Mask-R-CNN

# Appendix

# “Deconvolution” or Transposed Conv.

Want to swap the input and output dimensions