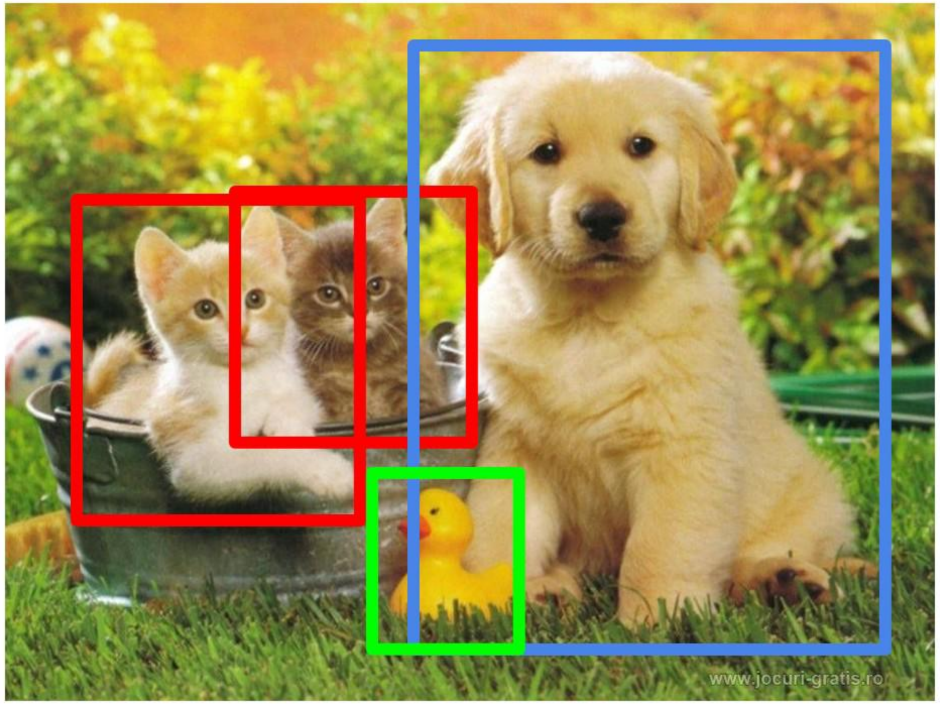
2019
怪兽
学堂

# Object detection

虾米

时间：2019-04

# Object Detection



# 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 Classification

**CAT? NO**

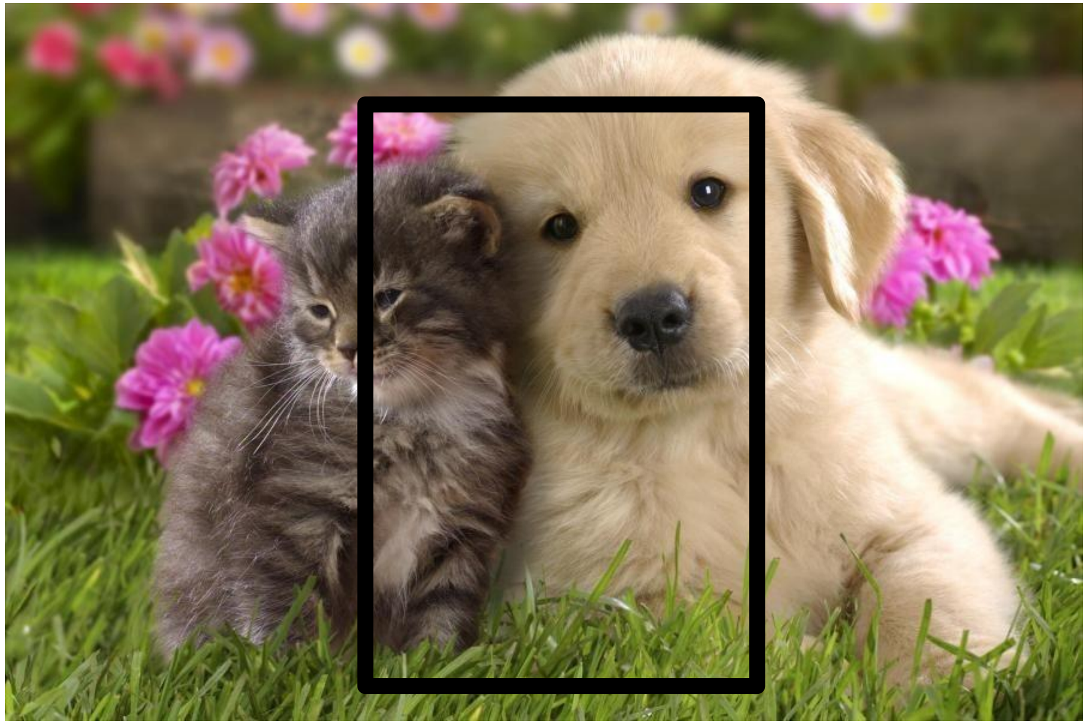
**DOG? NO**

# Detection as Classification

CAT? YES!

DOG? NO

# Detection as Classification



**CAT? NO**

**DOG? NO**

# Detection as Classification

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

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

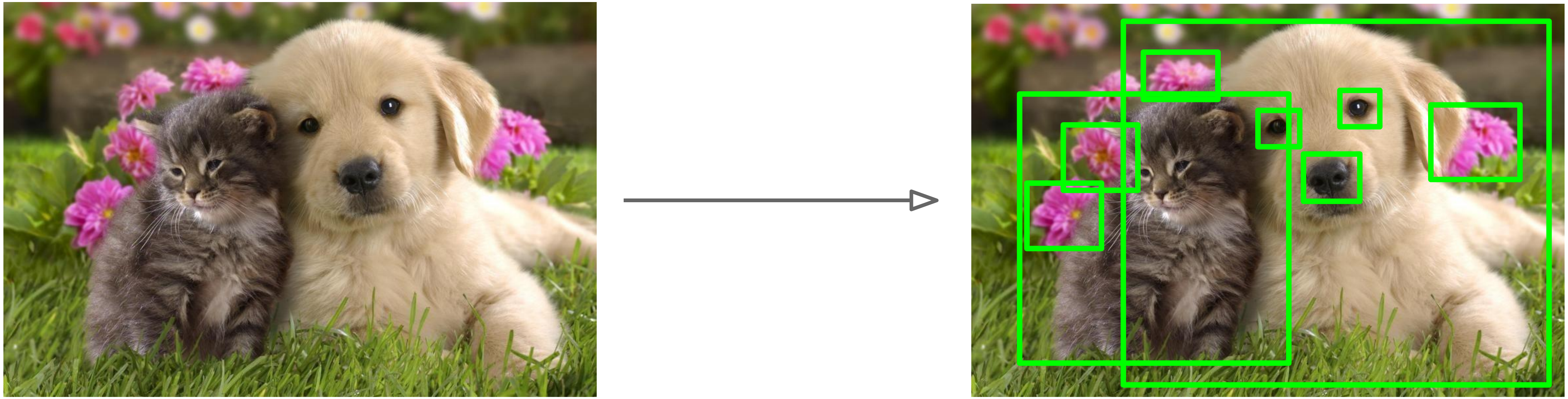
# Detection as Classification

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

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

# Region Proposals

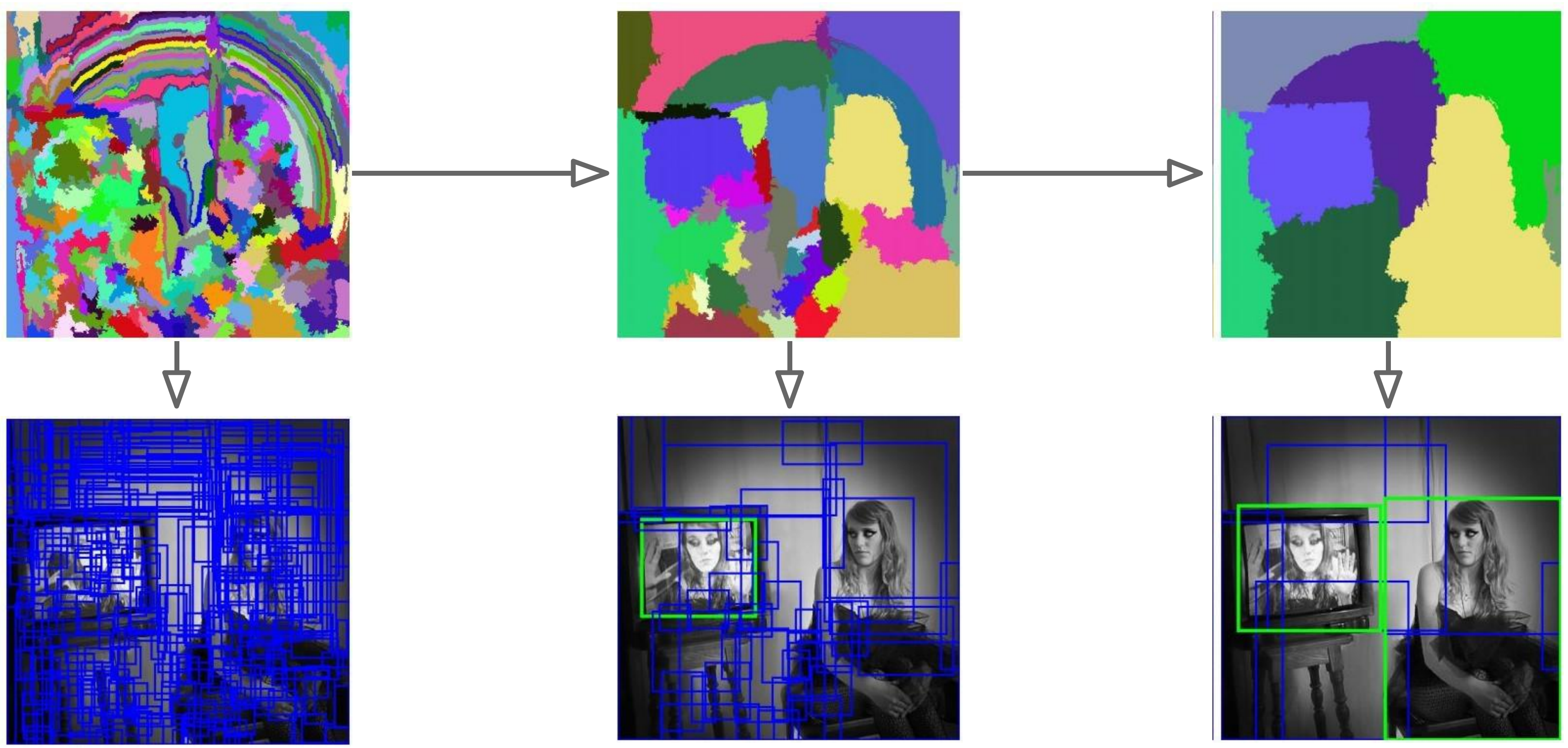
- Find “blobby” image regions that are likely to contain objects
- “Class-agnostic” object detector
- Look for “blob-like” regions



# Region Proposals: Selective Search

Bottom-up segmentation, merging regions at multiple scales

Convert regions to boxes



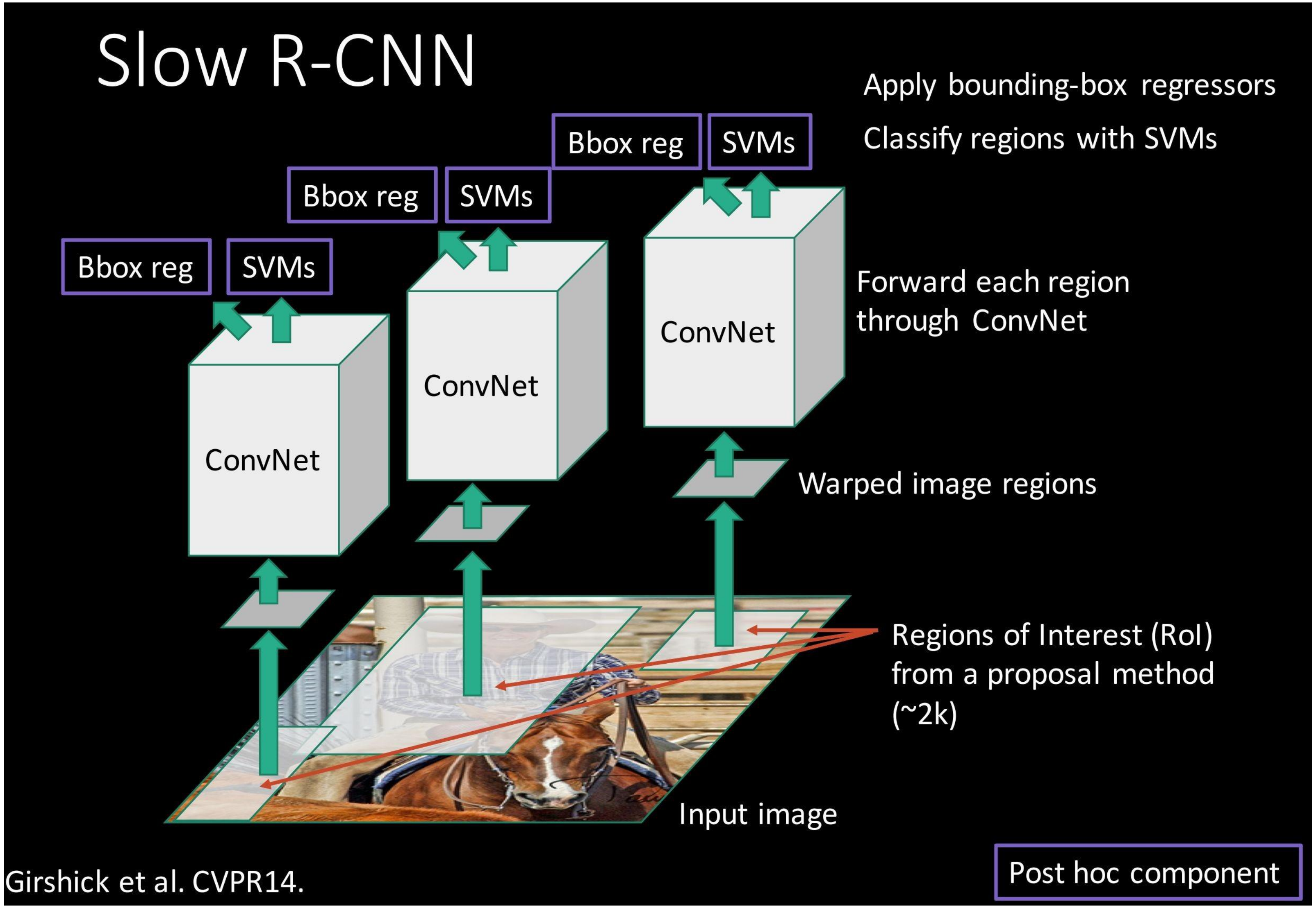
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013

# Region Proposals: Many other choices

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

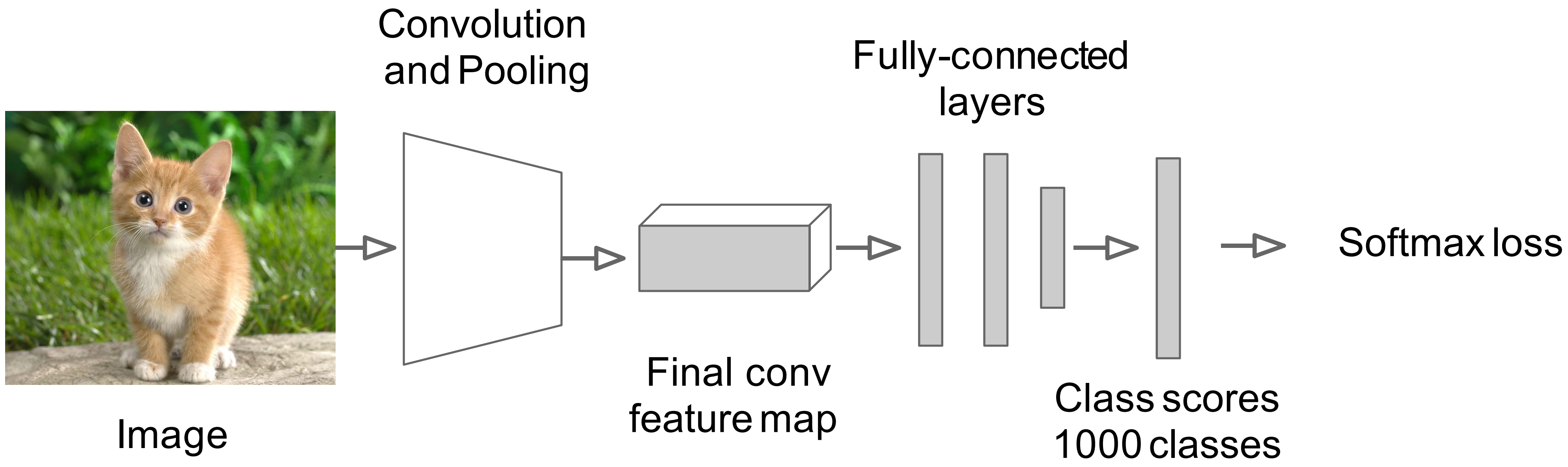
Hosang et al, "What makes for effective detection proposals?", PAMI 2015

# Putting it together: R-CNN



# R-CNN Training

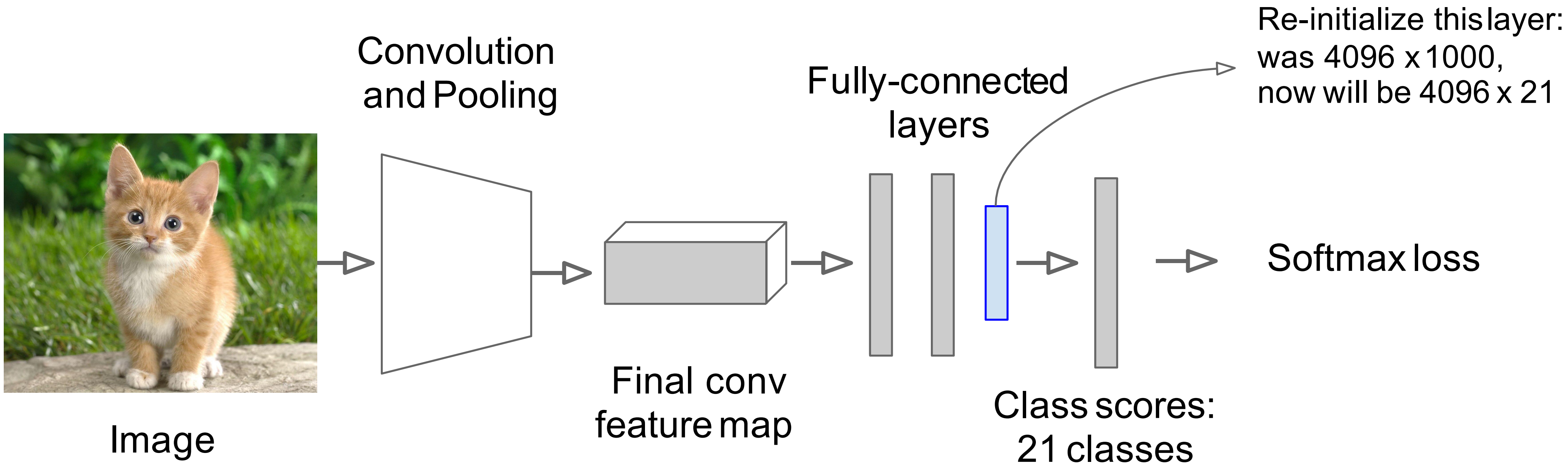
**Step 1**: Train (or download) a classification model for ImageNet (AlexNet)



# R-CNN Training

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

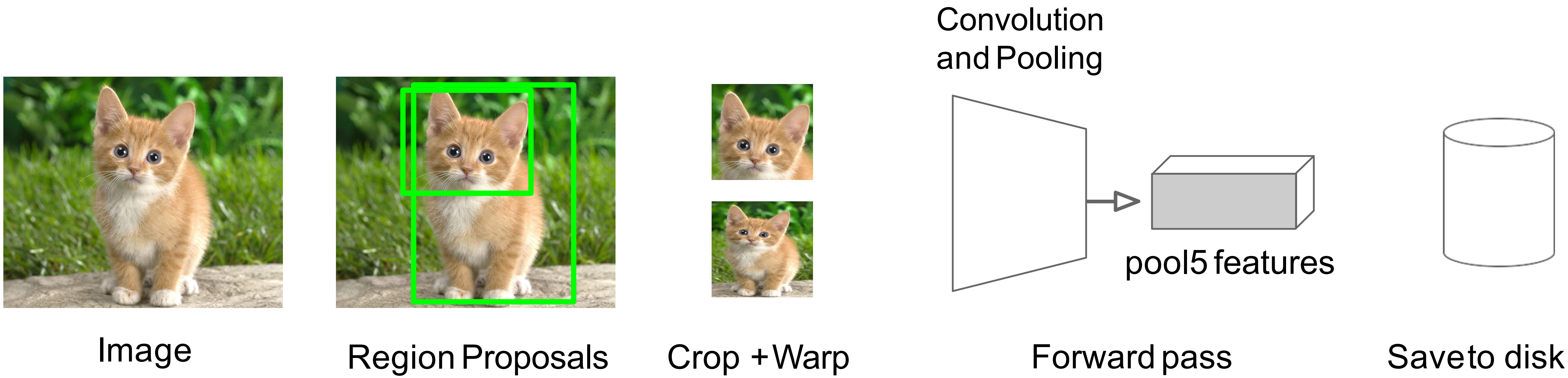
- Instead of 1000 ImageNet classes, want 20 object classes + background
- Throw away final fully-connected layer, reinitialize from scratch
- Keep training model using positive / negative regions from detection images



# R-CNN Training

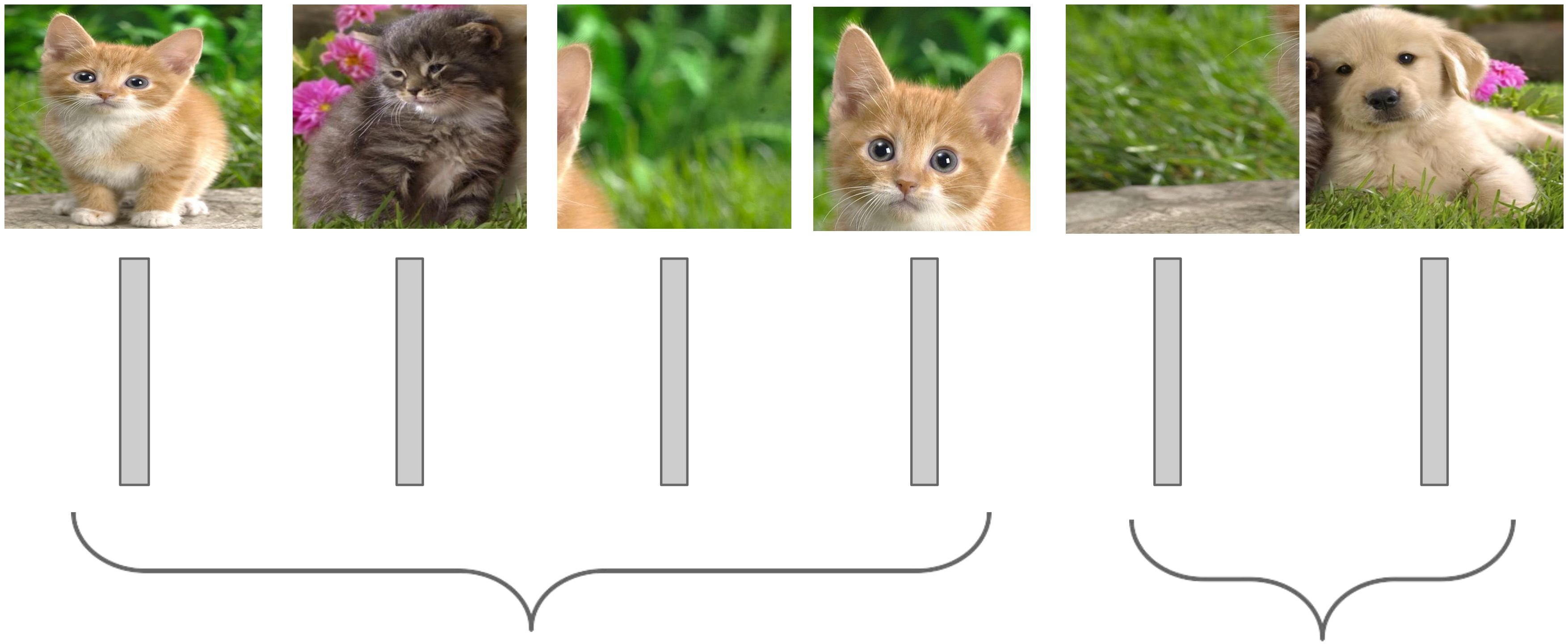
**Step 3**: Extract features

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



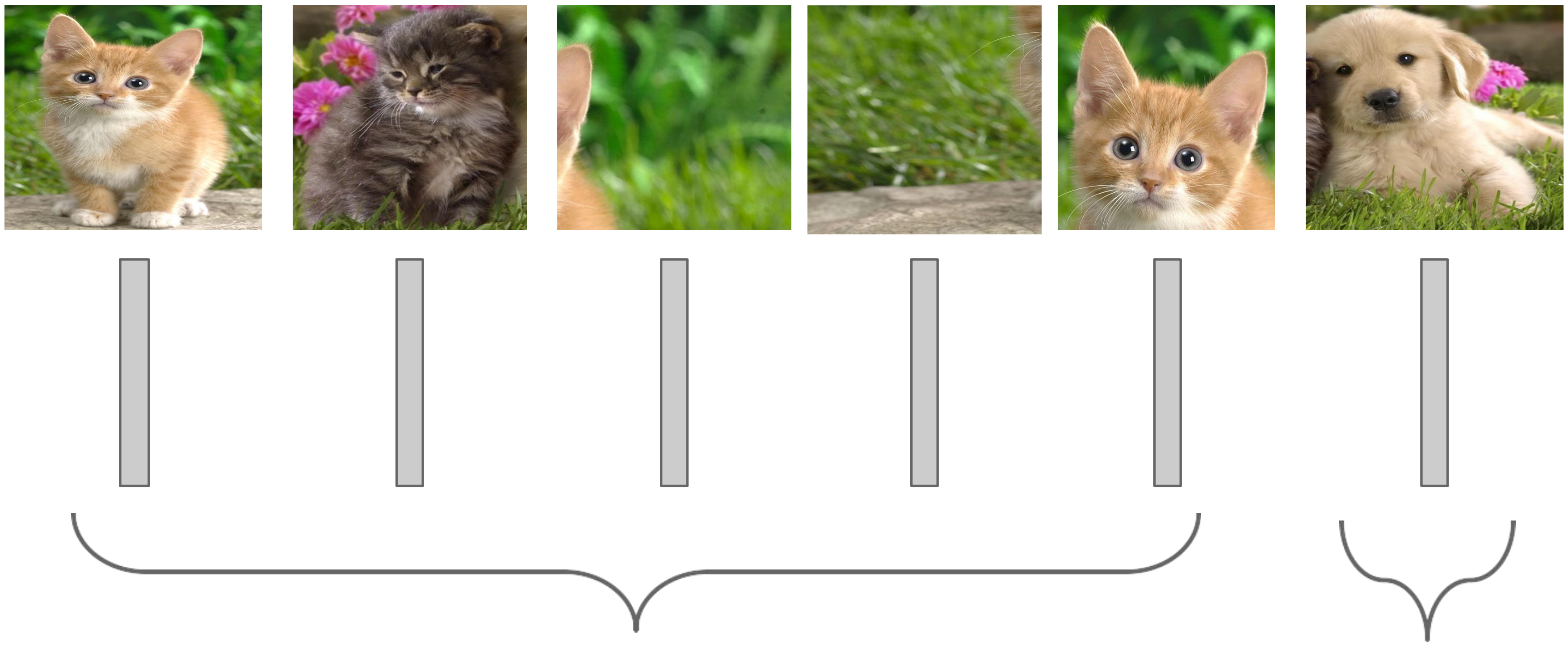
# R-CNN Training

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



# R-CNN Training

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



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)<br>Normalized coordinates | (0, 0, 0, 0)<br>Proposal is good | (.25, 0, 0, 0)<br>Proposal too far to left | (0, 0, -0.125, 0)<br>Proposal too wide |

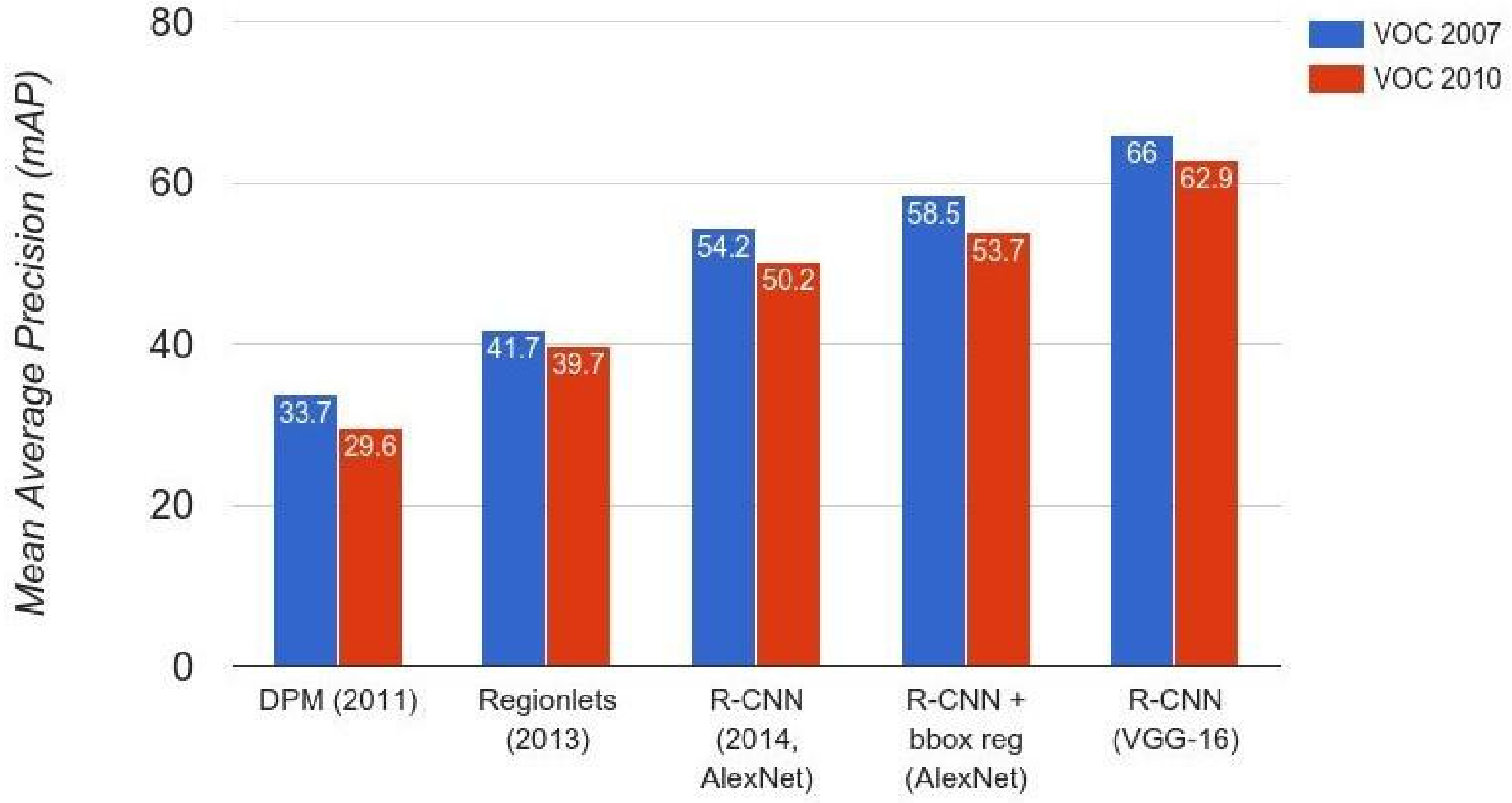
# Object Detection: Datasets

| | 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



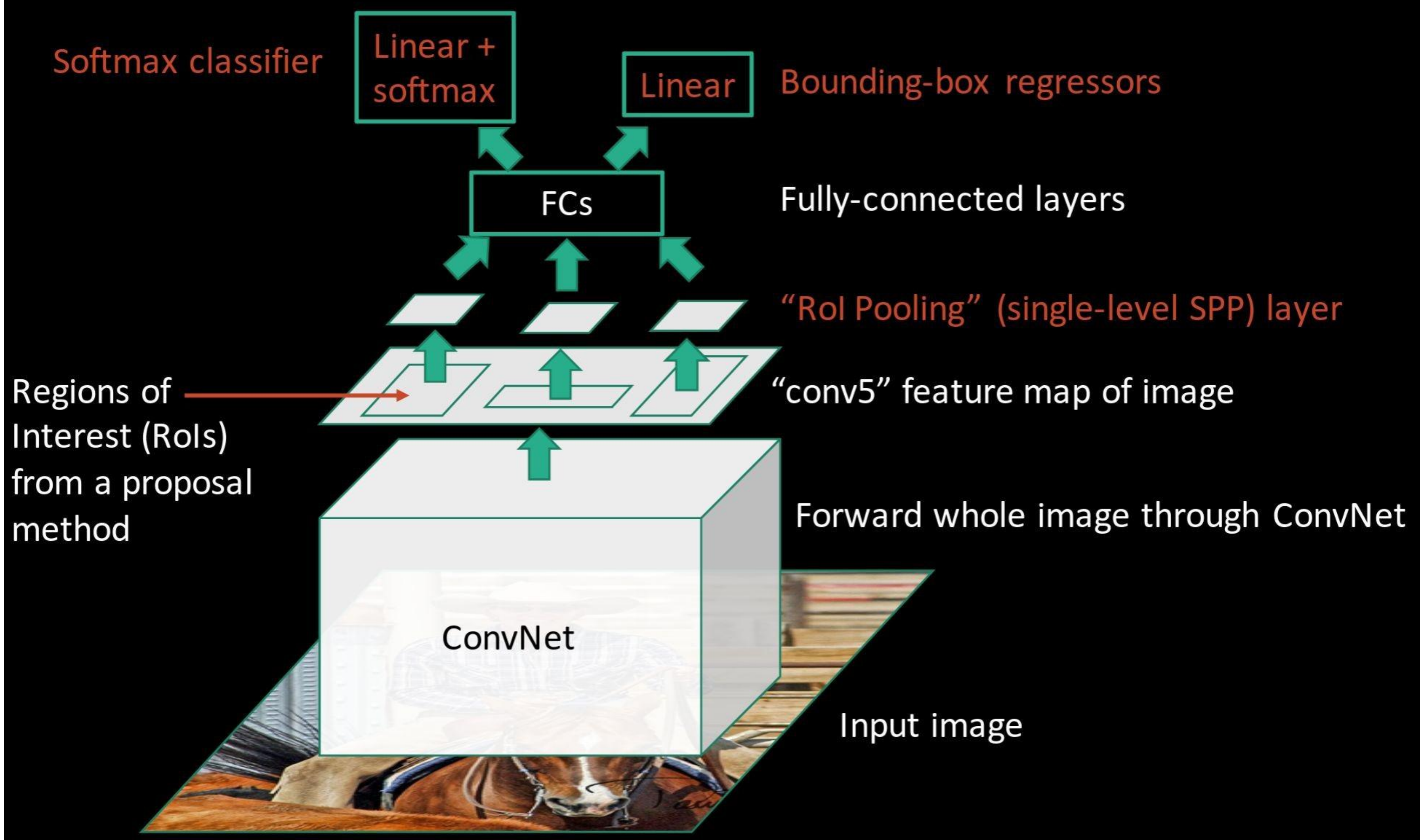
# R-CNN Problems

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

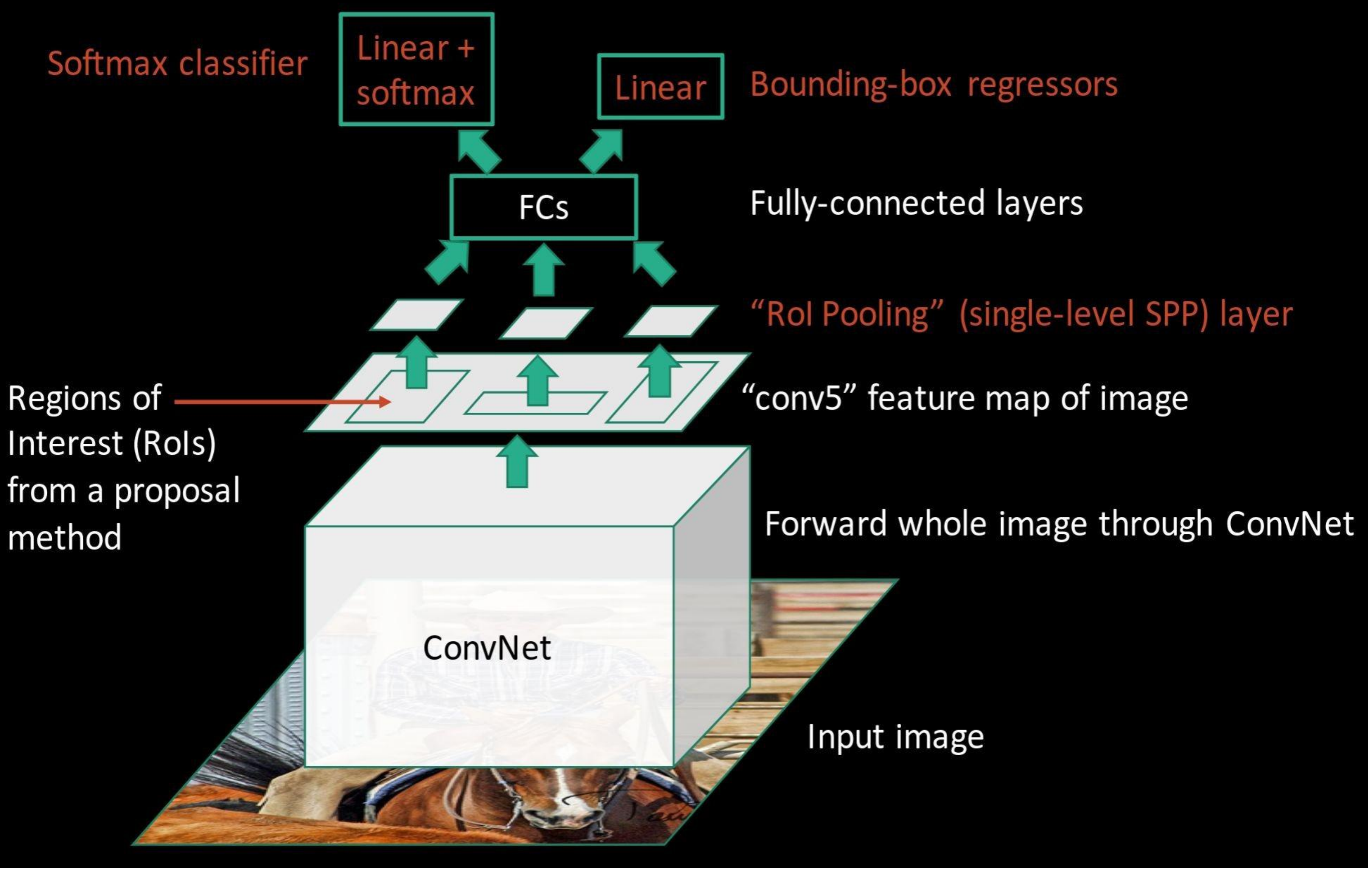
2. SVMs and regressors are post-hoc: CNN features not updated in response to SVMs and regressors

3. Complex multistage training pipeline

# Fast R-CNN (test time)

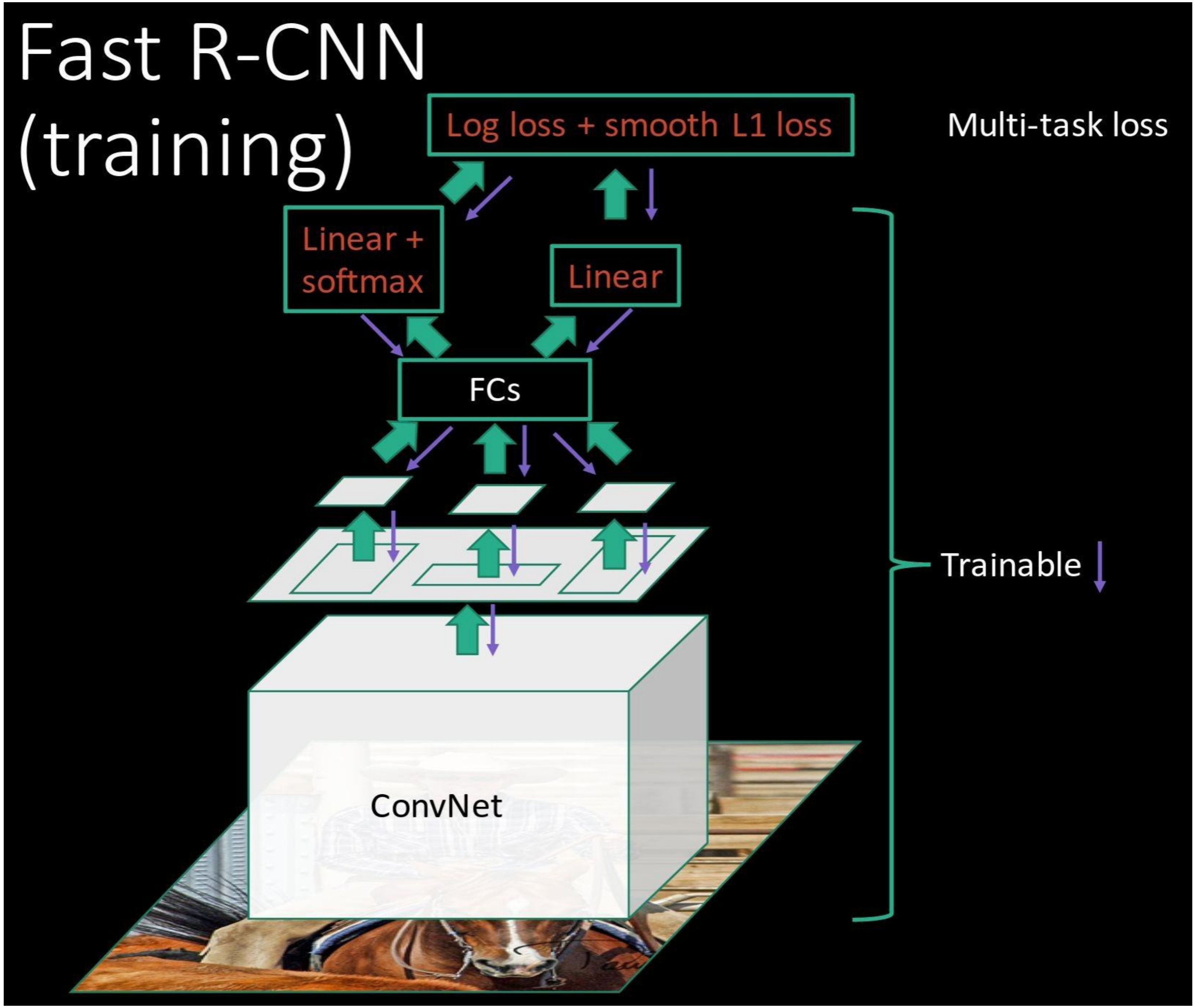


# Fast R-CNN (test time)




**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




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

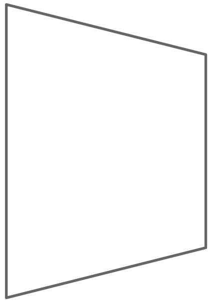
**R-CNN Problem #3:**
Complex training pipeline

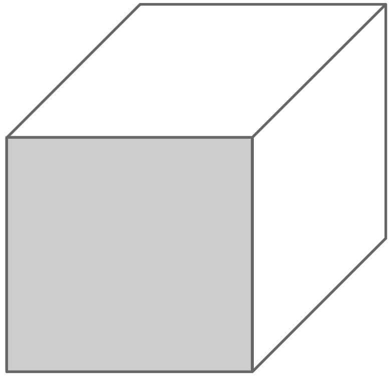
**Solution:**
Just train the whole system end-to-end all at once!

# Fast R-CNN: Region of Interest Pooling

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

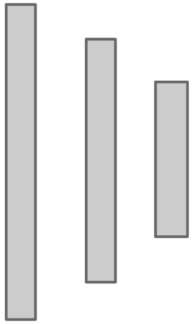
Convolution
and Pooling





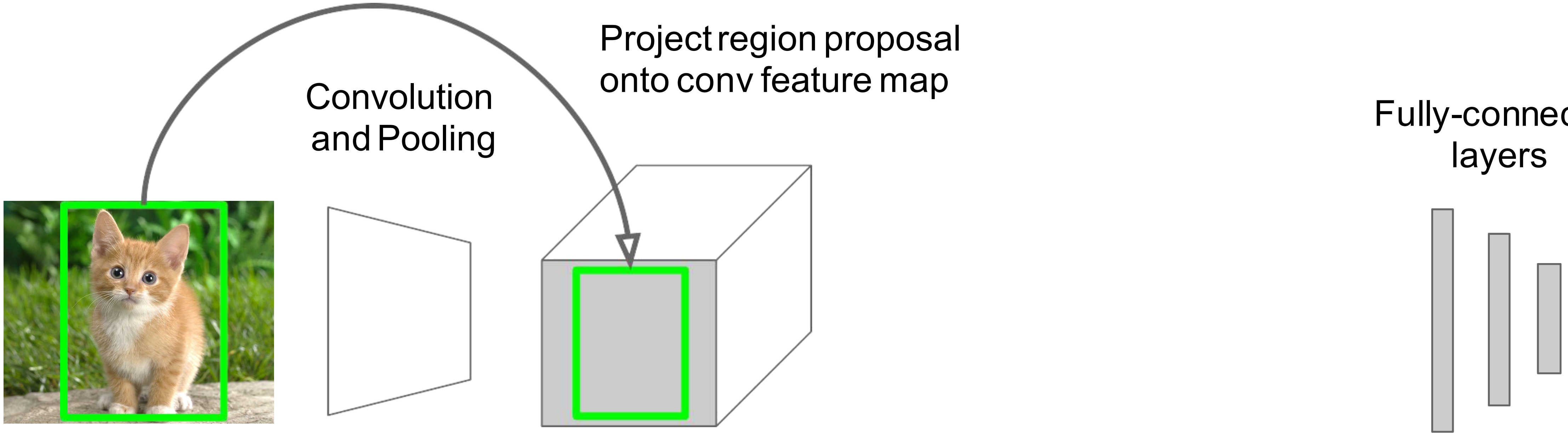
Hi-res conv features:
C x H x W
with region proposal

Fully-connected
layers



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

# Fast R-CNN: Region of Interest Pooling

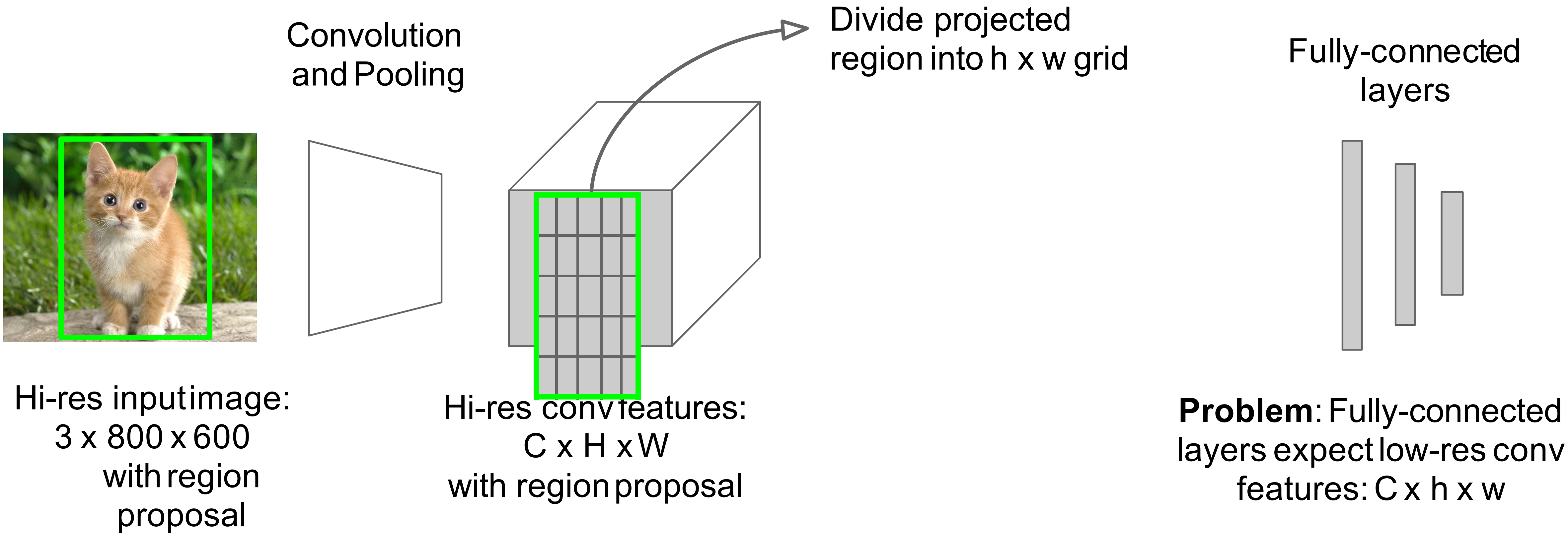



Hi-res input image:
3 x 800 x 600
with region
proposal

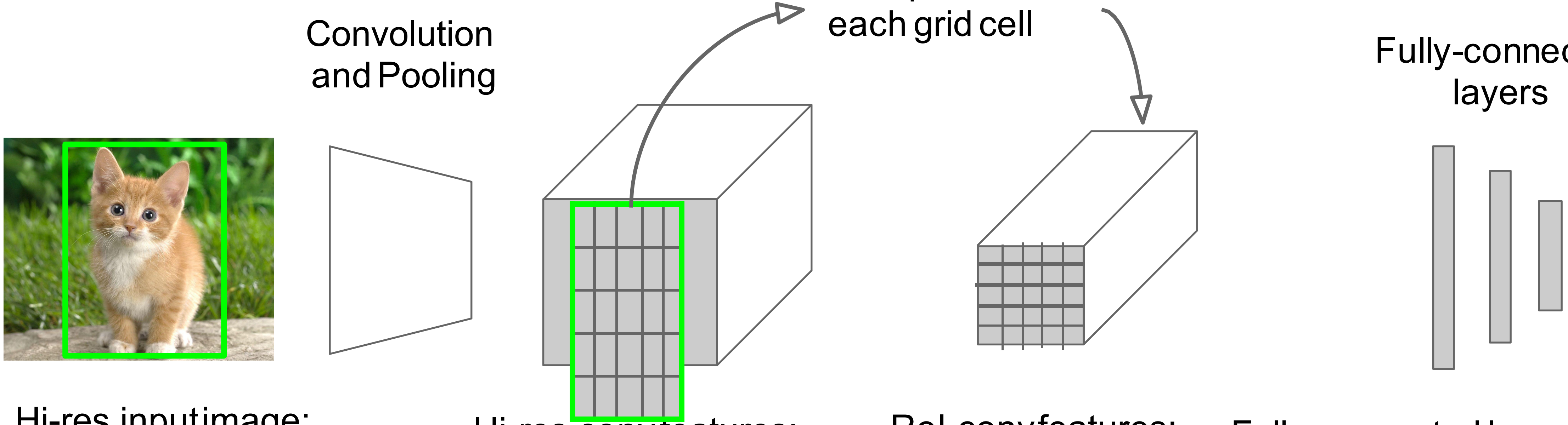
Hi-res conv features:
C x H x W
with region proposal

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

# Fast R-CNN: Region of Interest Pooling



# 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

| | | R-CNN | Fast R-CNN |
|---|---|---|---|
| Faster! | Training Time: | 84 hours | **9.5 hours** |
| | (Speedup) | 1x | **8.8x** |
| FASTER! | 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

| | | R-CNN | Fast R-CNN |
|---|---|---|---|
| Faster! | Training Time: | 84 hours | **9.5 hours** |
| | (Speedup) | 1x | **8.8x** |
| FASTER! | Test time per image | 47 seconds | **0.32 seconds** |
| | (Speedup) | 1x | **146x** |
| Better! | 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

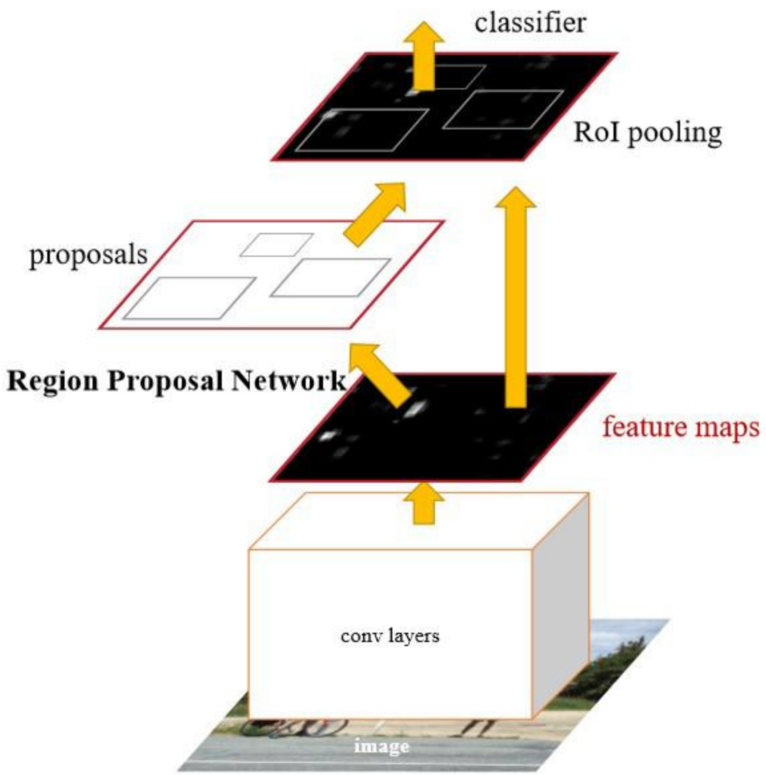
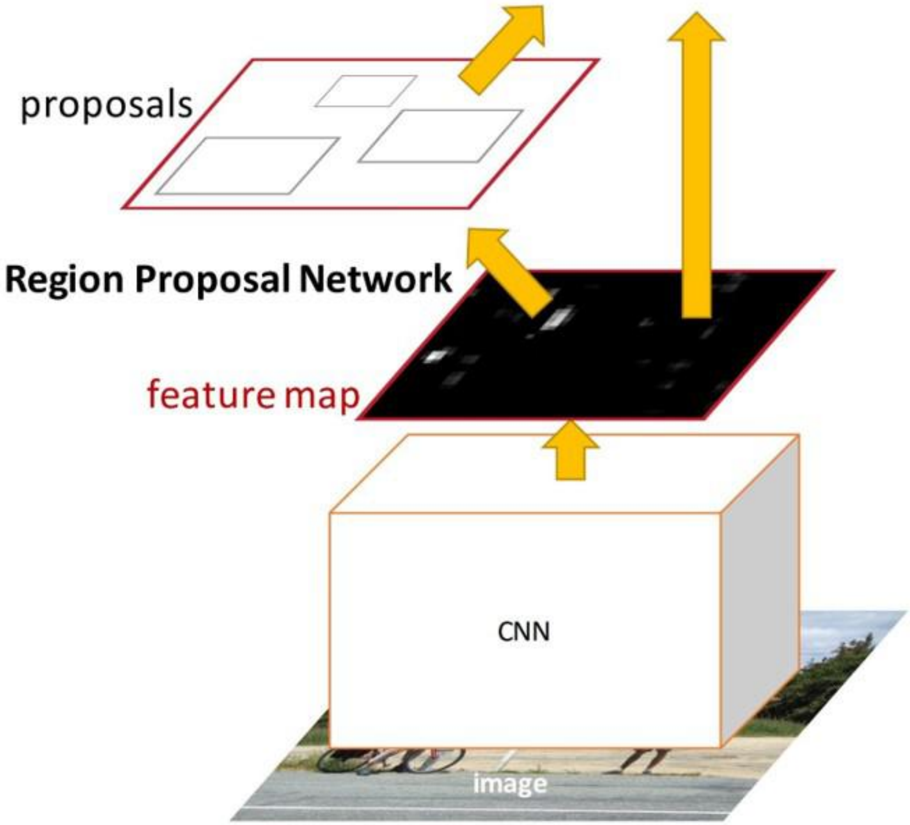


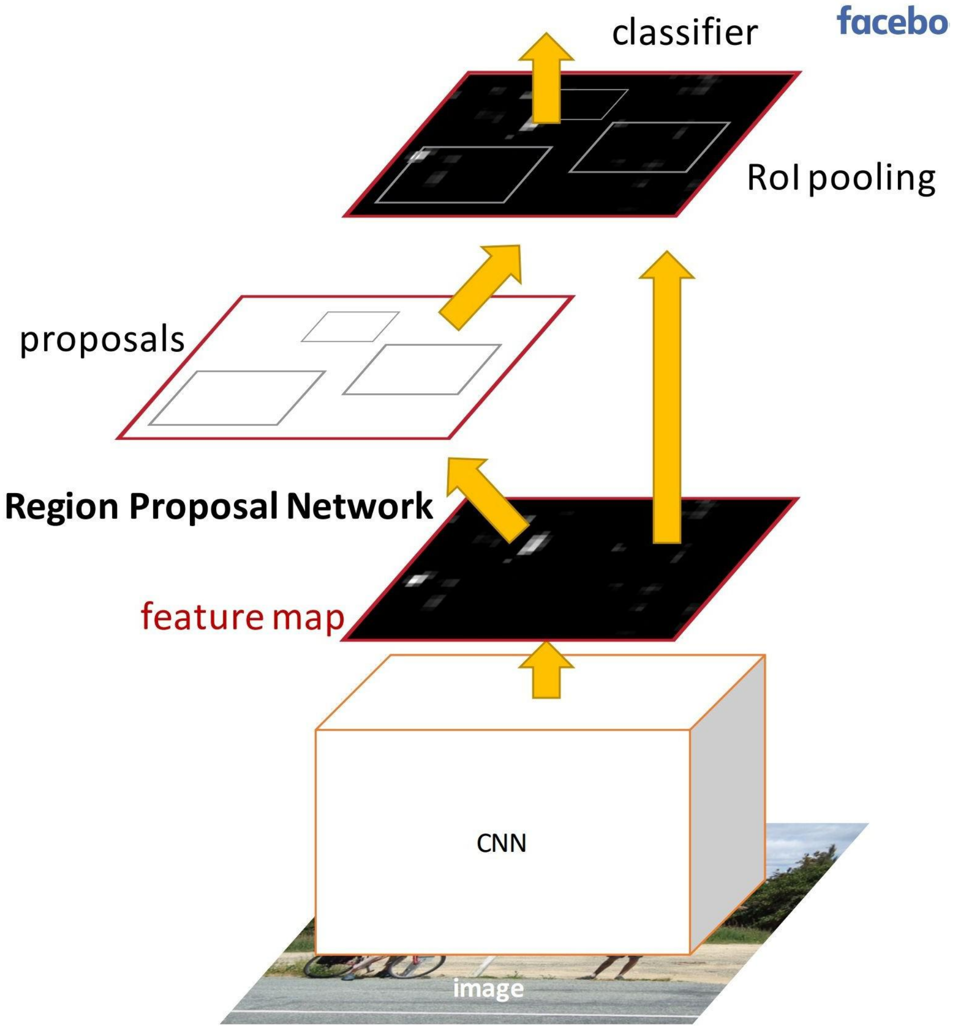

Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

# Region Proposal Networks(RPN)

- Input: Image(of any size)
- Output: A set of rectangular object proposals each with an objectness score
- Goal: share computation with a Fast R-CNN object detection network
- Model: fully-convolutional network



# 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

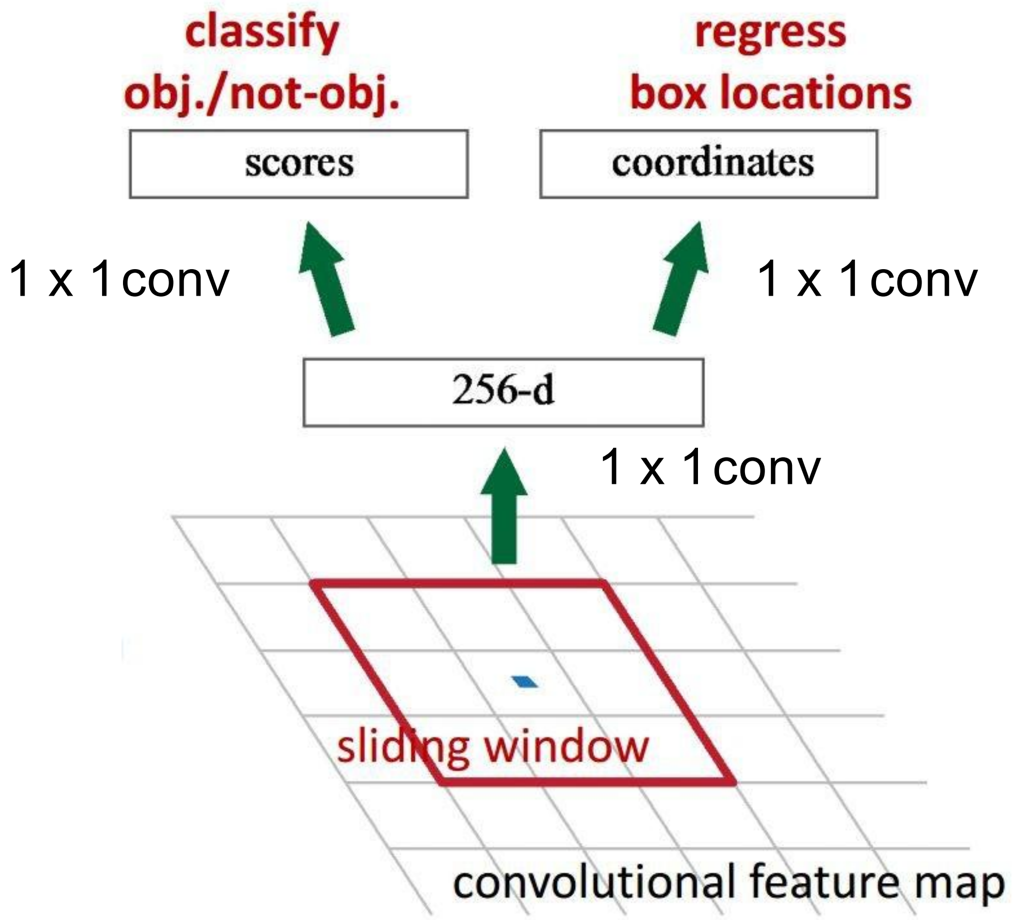
# Faster R-CNN: Region Proposal Network

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 windowprovides localization information with reference to the image

Box regression provides finer localization information with reference to this sliding window



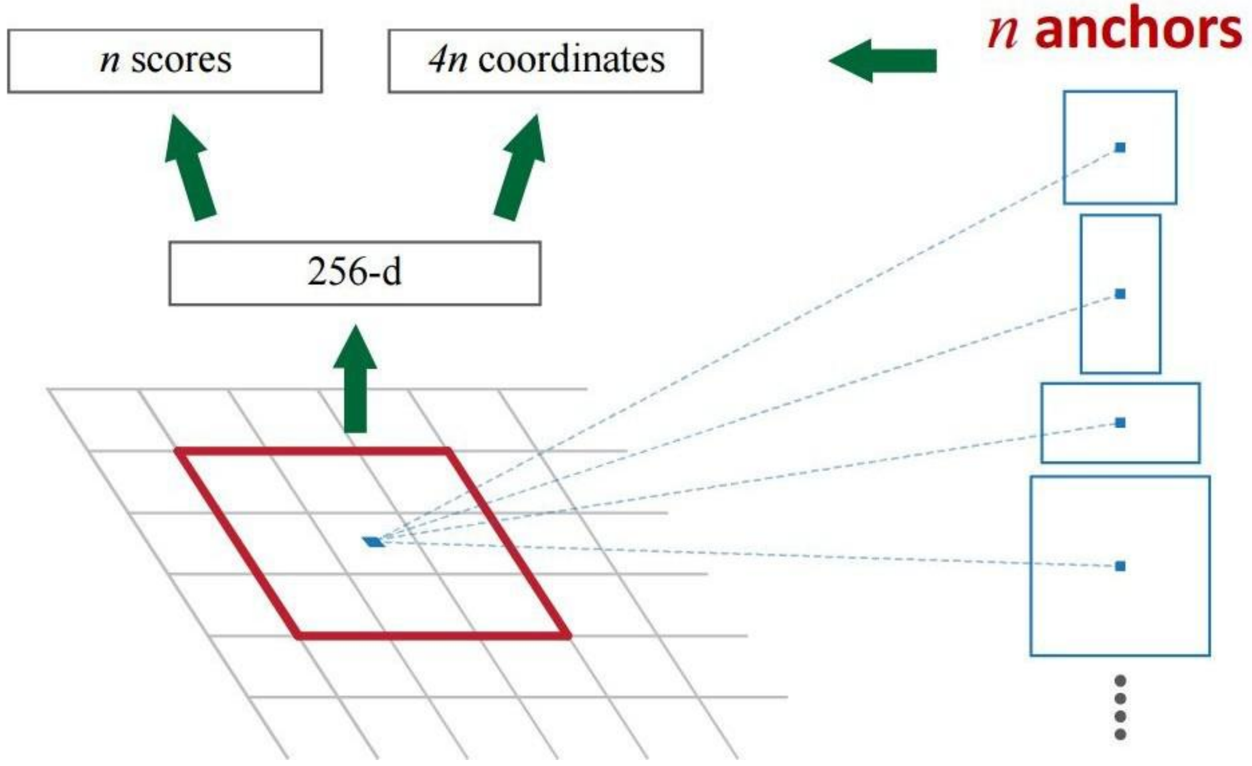
# Faster R-CNN: Region Proposal Network

Use **N anchor boxes** at each location

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

Regression gives offsets from anchor boxes

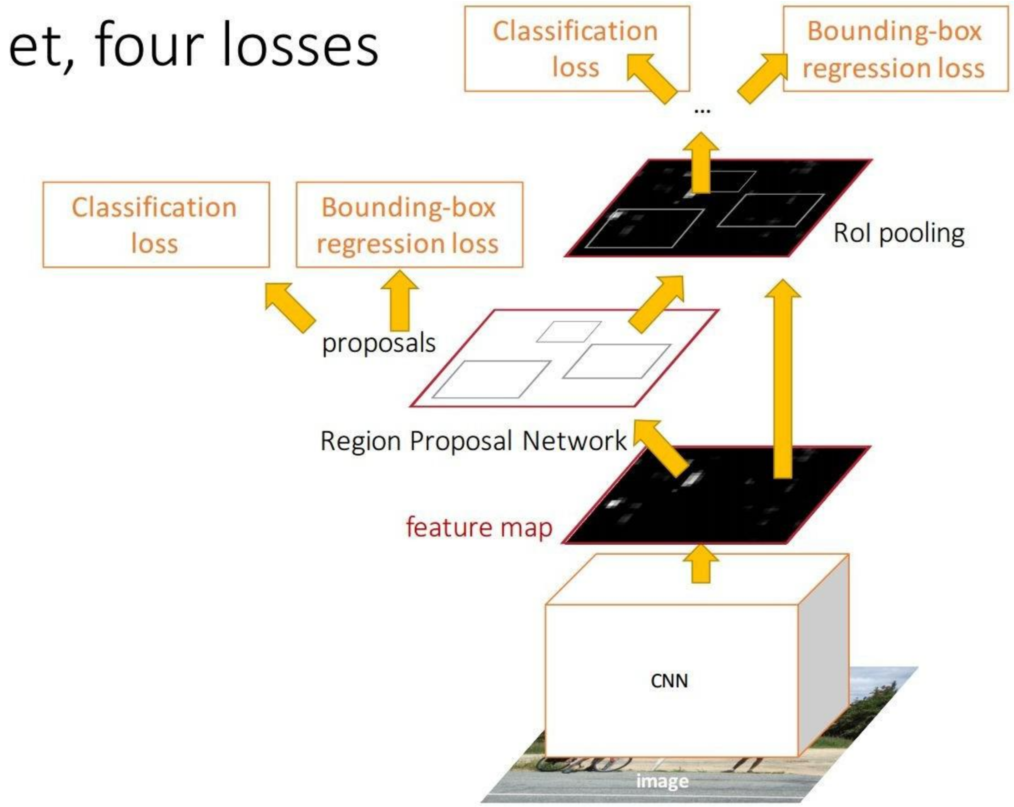
Classification gives the probability that each (regressed) anchor shows an object



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



# 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** |

# Object Detection code links:

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

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

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

2019
怪兽
学堂

# THANKS