Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

SHAOQING REN, KAIMING HE, ROSS GIRSHICK, JIAN SUN
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 Regression?

DOG, (x, y, w, h)
CAT, (x, y, w, h)

= 8 numbers
Detection as Regression?

CAT, \((x, y, w, h)\)
CAT, \((x, y, w, h)\)

\[\ldots\]
CAT \((x, y, w, h)\)

= many numbers

Need variable sized outputs
Detection as Classification

CAT? NO
DOG? NO
Detection as Classification

CAT? YES!

DOG? NO
Detection as Classification

CAT? NO
DOG? NO
Detection as Classification

Problem: Need to test many positions and scales

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

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

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

Region Proposals: Many other choices

<table>
<thead>
<tr>
<th>Method</th>
<th>Approach</th>
<th>Outputs Segments</th>
<th>Outputs Score</th>
<th>Control #proposals</th>
<th>Time (sec.)</th>
<th>Repeatability</th>
<th>Recall Results</th>
<th>Detection Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bing [18]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.2</td>
<td>***</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>CPMC [19]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>250</td>
<td>-</td>
<td>**</td>
<td>*</td>
</tr>
<tr>
<td>EdgeBoxes [20]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>0.3</td>
<td>**</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Endres [21]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>100</td>
<td>-</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>Geodesic [22]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>***</td>
<td>**</td>
</tr>
<tr>
<td>MCG [23]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>30</td>
<td>*</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Objectness [24]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Rahtu [25]</td>
<td>Window scoring</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>3</td>
<td>-</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>RandomizedPrim’s [26]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>Rantalankila [27]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>-</td>
<td>**</td>
</tr>
<tr>
<td>Rigor [28]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>*</td>
<td>**</td>
<td>**</td>
</tr>
<tr>
<td>SelectiveSearch [29]</td>
<td>Grouping</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>10</td>
<td>**</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Gaussian</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>SlidingWindow</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td>***</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Superpixels</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>1</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Uniform</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

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

![Diagram of R-CNN model](image)

- Image
- Convolution and Pooling
- Final conv feature map
- Fully-connected layers
- Class scores: 21 classes
- Softmax loss

Re-initialize this layer: was 4096 x 1000, now will be 4096 x 21
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

- **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

- 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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Number of classes</strong></td>
<td>20</td>
<td>200</td>
<td>80</td>
</tr>
<tr>
<td><strong>Number of images (train + val)</strong></td>
<td>~20k</td>
<td>~470k</td>
<td>~120k</td>
</tr>
<tr>
<td><strong>Mean objects per image</strong></td>
<td>2.4</td>
<td>1.1</td>
<td>7.2</td>
</tr>
</tbody>
</table>
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 Results Graph](image)

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

- Softmax classifier
- Linear + softmax
- Linear
- Bounding-box regressors
- Fully-connected layers
- “RoI Pooling” (single-level SPP) layer
- “conv5” feature map of image
- Forward whole image through ConvNet

Regions of Interest (RoIs) from a proposal method

ConvNet

Input image

Slide credit: Ross Girshick
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 Girshick
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

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

Fully-connected
layers

Fei-Fei Li & Andrej Karpathy & Justin Johnson
Lecture 8 - 70
1 Feb 2016
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

Convolution and Pooling

Project region proposal onto conv feature map

Fully-connected layers
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

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

Fully-connected layers
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
- Max-pool within each grid cell
- RoI conv features: C x h x w for region proposal
- Fully-connected layers expect low-res conv features: C x h x w

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 8 - 73
1 Feb 2016
Fast R-CNN Results

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time:</td>
<td>84 hours</td>
<td>9.5 hours</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>8.8x</td>
</tr>
</tbody>
</table>

Faster!

Using VGG-16 CNN on Pascal VOC 2007 dataset
## Fast R-CNN Results

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training Time:</td>
<td>84 hours</td>
<td>9.5 hours</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>8.8x</td>
</tr>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
Fast R-CNN Results

<table>
<thead>
<tr>
<th>Faster!</th>
<th>Training Time:</th>
<th>84 hours</th>
<th>9.5 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(Speedup)</td>
<td>1x</td>
<td>8.8x</td>
</tr>
<tr>
<td></td>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td></td>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
<tr>
<td>FASTER!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Better!</td>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
</tr>
</tbody>
</table>

Using VGG-16 CNN on Pascal VOC 2007 dataset
Fast R-CNN Problem:

Test-time speeds don’t include region proposals

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
<tr>
<td>Test time per image with Selective Search</td>
<td>50 seconds</td>
<td>2 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
</tr>
</tbody>
</table>
Fast R-CNN Problem Solution:

Test-time speeds don’t include region proposals
Just make the CNN do region proposals too!

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image</td>
<td>47 seconds</td>
<td>0.32 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>146x</td>
</tr>
<tr>
<td>Test time per image</td>
<td>50 seconds</td>
<td>2 seconds</td>
</tr>
<tr>
<td>with Selective Search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
</tr>
</tbody>
</table>
Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the ‘attention’ of this unified network.


Slide credit: Ross Girshick
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


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
Loss Function

\[ L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \]
### Experiments

Table 1: Detection results on **PASCAL VOC 2007 test set** (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF, but using various proposal methods for training and testing.

<table>
<thead>
<tr>
<th>train-time region proposals</th>
<th>test-time region proposals</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>method</td>
<td># boxes</td>
<td>method</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>SS</td>
</tr>
<tr>
<td>EB</td>
<td>2k</td>
<td>EB</td>
</tr>
<tr>
<td>RPN+ZF, shared</td>
<td>2k</td>
<td>RPN+ZF, shared</td>
</tr>
</tbody>
</table>

*ablation experiments follow below*

<table>
<thead>
<tr>
<th>RPN+ZF, unshared</th>
<th>2k</th>
<th>RPN+ZF, unshared</th>
<th>300</th>
<th>58.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF</td>
<td>100</td>
<td>55.1</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF</td>
<td>300</td>
<td>56.8</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF</td>
<td>1k</td>
<td>56.3</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF (no NMS)</td>
<td>6k</td>
<td>55.2</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF (no cls)</td>
<td>100</td>
<td>44.6</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF (no cls)</td>
<td>300</td>
<td>51.4</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF (no cls)</td>
<td>1k</td>
<td>55.8</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF (no reg)</td>
<td>300</td>
<td>52.1</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+ZF (no reg)</td>
<td>1k</td>
<td>51.3</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>RPN+VGG</td>
<td>300</td>
<td>59.2</td>
</tr>
</tbody>
</table>
Table 2: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: “07”: VOC 2007 trainval, “07+12”: union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2k. †: this was reported in [5]; using the repository provided by this paper, this number is higher (68.0±0.3 in six runs).

<table>
<thead>
<tr>
<th>method</th>
<th># proposals</th>
<th>data</th>
<th>mAP (%)</th>
<th>time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2k</td>
<td>07</td>
<td>66.9†</td>
<td>1830</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>07+12</td>
<td>70.0</td>
<td>1830</td>
</tr>
<tr>
<td>RPN+VGG, unshared</td>
<td>300</td>
<td>07</td>
<td>68.5</td>
<td>342</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07</td>
<td>69.9</td>
<td>198</td>
</tr>
<tr>
<td>RPN+VGG, shared</td>
<td>300</td>
<td>07+12</td>
<td>73.2</td>
<td>198</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>method</th>
<th># proposals</th>
<th>data</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>2k</td>
<td>12</td>
<td>65.7</td>
</tr>
<tr>
<td>SS</td>
<td>2k</td>
<td>07++12</td>
<td>68.4</td>
</tr>
<tr>
<td>RPN+VGG, shared†</td>
<td>300</td>
<td>12</td>
<td>67.0</td>
</tr>
<tr>
<td>RPN+VGG, shared‡</td>
<td>300</td>
<td>07++12</td>
<td>70.4</td>
</tr>
</tbody>
</table>
Table 4: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. “Region-wise” includes NMS, pooling, fc, and softmax. See our released code for the profiling of running time.

<table>
<thead>
<tr>
<th>model</th>
<th>system</th>
<th>conv</th>
<th>proposal</th>
<th>region-wise</th>
<th>total</th>
<th>rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG</td>
<td>SS + Fast R-CNN</td>
<td>146</td>
<td>1510</td>
<td>174</td>
<td>1830</td>
<td>0.5 fps</td>
</tr>
<tr>
<td>VGG</td>
<td>RPN + Fast R-CNN</td>
<td>141</td>
<td>10</td>
<td>47</td>
<td>198</td>
<td>5 fps</td>
</tr>
<tr>
<td>ZF</td>
<td>RPN + Fast R-CNN</td>
<td>31</td>
<td>3</td>
<td>25</td>
<td>59</td>
<td>17 fps</td>
</tr>
</tbody>
</table>
Experiments

Figure 2: Recall vs. IoU overlap ratio on the PASCAL VOC 2007 test set.
Table 5: **One-Stage Detection vs. Two-Stage Proposal + Detection.** Detection results are on the PASCAL VOC 2007 test set using the ZF model and Fast R-CNN. RPN uses unshared features.

<table>
<thead>
<tr>
<th></th>
<th>regions</th>
<th>detector</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two-Stage</td>
<td>RPN + ZF, unshared</td>
<td>Fast R-CNN + ZF, 1 scale</td>
<td>58.7</td>
</tr>
<tr>
<td>One-Stage</td>
<td>dense, 3 scales, 3 asp. ratios</td>
<td>Fast R-CNN + ZF, 1 scale</td>
<td>53.8</td>
</tr>
<tr>
<td>One-Stage</td>
<td>dense, 3 scales, 3 asp. ratios</td>
<td>Fast R-CNN + ZF, 5 scales</td>
<td>53.9</td>
</tr>
</tbody>
</table>
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 Girshick
## Faster R-CNN: Results

<table>
<thead>
<tr>
<th></th>
<th>R-CNN</th>
<th>Fast R-CNN</th>
<th>Faster R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test time per image (with proposals)</td>
<td>50 seconds</td>
<td>2 seconds</td>
<td>0.2 seconds</td>
</tr>
<tr>
<td>(Speedup)</td>
<td>1x</td>
<td>25x</td>
<td>250x</td>
</tr>
<tr>
<td>mAP (VOC 2007)</td>
<td>66.0</td>
<td>66.9</td>
<td>66.9</td>
</tr>
</tbody>
</table>
Object Detection State-of-the-art:
ResNet 101 + Faster R-CNN + some extras

<table>
<thead>
<tr>
<th>training data</th>
<th>COCO train</th>
<th>COCO trainval</th>
</tr>
</thead>
<tbody>
<tr>
<td>test data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>baseline Faster R-CNN (VGG-16)</td>
<td>41.5</td>
<td>21.2</td>
</tr>
<tr>
<td>baseline Faster R-CNN (ResNet-101) +box refinement</td>
<td>48.4</td>
<td>27.2</td>
</tr>
<tr>
<td>+context</td>
<td>51.1</td>
<td>30.0</td>
</tr>
<tr>
<td>+multi-scale testing</td>
<td>53.8</td>
<td>32.5</td>
</tr>
</tbody>
</table>

ImageNet Detection 2013 - 2015

ImageNet Detection (mAP)

- NeoNet ensemble (2015): 53.57
- Faster R-CNN single (2015): 42.94
- GoogleNet ensemble (2014): 43.93
- NUS ensemble (2014): 37.21
- SPP ensemble (2014): 35.11
- UvA-Euvison (2013): 22.58
- Overfeat (2013): 19.4
Object Detection code links:

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

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

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

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
QUESTIONS???
THANK YOU😊