Object detection
The Task
R-CNN: Regions with CNN features

Input image 
Extract region proposals (~2k / image) 
Compute CNN features 
Classify regions (linear SVM)

Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation

R. Girshick, J. Donahue, T. Darrell, J. Malik
IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014
R-CNN at test time: Step 2

Input image → Extract region proposals (~2k / image) → Compute CNN features

Slide credit: Ross Girshick
R-CNN at test time: Step 2

Input image

Extract region proposals (~2k / image)

Compute CNN features

Slide credit: Ross Girshick
R-CNN at test time: Step 2

1. Crop
2. Scale (anisotropic)
3. Forward propagate

Output: “fc7” features

Input image
Extract region proposals (~2k / image)
Compute CNN features

Slide credit: Ross Girshick
R-CNN at test time: Step 3

Input image

Extract region proposals (~2k / image)

Compute CNN features

Classify regions

Warped proposal

4096-dimensional fc7 feature vector

linear classifiers (SVM or softmax)

Slide credit: Ross Girshick
Step 4: Object proposal refinement

Original proposal → Linear regression on CNN features → Predicted object bounding box

Bounding-box regression

Slide credit: Ross Girshick
# R-CNN results on PASCAL

<table>
<thead>
<tr>
<th>System</th>
<th>VOC 2007</th>
<th>VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 (Girshick et al. 2011)</td>
<td>33.7%</td>
<td>29.6%</td>
</tr>
<tr>
<td>UVA sel. search (Uijlings et al. 2013)</td>
<td></td>
<td>35.1%</td>
</tr>
<tr>
<td>Regionlets (Wang et al. 2013)</td>
<td>41.7%</td>
<td>39.7%</td>
</tr>
<tr>
<td>SegDPM (Fidler et al. 2013)</td>
<td></td>
<td>40.4%</td>
</tr>
</tbody>
</table>

**Reference systems**

source: Ross Girshick
## R-CNN results on PASCAL

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC 2007</th>
<th>VOC 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPM v5 (Girshick et al. 2011)</td>
<td>33.7%</td>
<td>29.6%</td>
</tr>
<tr>
<td>UVA sel. search (Uijlings et al. 2013)</td>
<td></td>
<td>35.1%</td>
</tr>
<tr>
<td>Regionlets (Wang et al. 2013)</td>
<td>41.7%</td>
<td>39.7%</td>
</tr>
<tr>
<td>SegDPM (Fidler et al. 2013)</td>
<td></td>
<td>40.4%</td>
</tr>
<tr>
<td>R-CNN</td>
<td>54.2%</td>
<td>50.2%</td>
</tr>
<tr>
<td>R-CNN + bbox regression</td>
<td>58.5%</td>
<td>53.7%</td>
</tr>
</tbody>
</table>

Metric: mean average precision (higher is better)

Slide credit: Ross Girshick
Training R-CNN

• Train convolutional network on ImageNet classification

• *Finetune* on detection
  • Classification problem!
  • Proposals with IoU > 50% are positives
  • Sample fixed proportion of positives in each batch because of imbalance
Speeding up R-CNN
Speeding up R-CNN
ROI Pooling

• How do we crop from a feature map?
• Step 1: Resize boxes to account for subsampling

Fast R-CNN. Ross Girshick. In ICCV 2015
ROI Pooling

• How do we crop from a feature map?
• Step 2: Snap to feature map grid
ROI Pooling

- How do we crop from a feature map?
- Step 3: Place a grid of fixed size
ROI Pooling

- How do we crop from a feature map?
- Step 4: Take max in each cell
Fast R-CNN

<table>
<thead>
<tr>
<th></th>
<th>Fast R-CNN</th>
<th>R-CNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train time (h)</td>
<td>9.5</td>
<td>84</td>
</tr>
<tr>
<td>Speedup</td>
<td>8.8x</td>
<td>1x</td>
</tr>
<tr>
<td>Test time / image</td>
<td>0.32s</td>
<td>47.0s</td>
</tr>
<tr>
<td>Speedup</td>
<td>146x</td>
<td>1x</td>
</tr>
<tr>
<td>mean AP</td>
<td>66.9</td>
<td>66.0</td>
</tr>
</tbody>
</table>
Fast R-CNN

• Bottleneck remaining (not included in time):
  • Object proposal generation

• Slow
  • Requires segmentation
  • $O(1s)$ per image
Faster R-CNN

• Can we produce object proposals from convolutional networks?

• A change in intuition
  • Instead of using grouping
  • Recognize likely objects?

• For every possible box, score if it is likely to correspond to an object

Faster R-CNN
Faster R-CNN

• At each location, consider boxes of many different sizes and aspect ratios
Faster R-CNN

- At each location, consider boxes of many different sizes and aspect ratios
Faster R-CNN

- At each location, consider boxes of many different sizes.
Faster R-CNN

• \( s \) scales * \( a \) aspect ratios = \( sa \) anchor boxes
• Use convolutional layer on top of filter map to produce \( sa \) scores
• Pick top few boxes as proposals
Faster R-CNN

<table>
<thead>
<tr>
<th>Method</th>
<th>mean AP (PASCAL VOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast R-CNN</td>
<td>65.7</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>67.0</td>
</tr>
</tbody>
</table>
## Impact of Feature Extractors

<table>
<thead>
<tr>
<th>ConvNet</th>
<th>mean AP (PASCAL VOC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG</td>
<td>70.4</td>
</tr>
<tr>
<td>ResNet 101</td>
<td>73.8</td>
</tr>
</tbody>
</table>
## Impact of Additional Data

<table>
<thead>
<tr>
<th>Method</th>
<th>Training data</th>
<th>mean AP (PASCAL VOC 2012 Test)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast R-CNN</td>
<td>VOC 12 Train (10K)</td>
<td>65.7</td>
</tr>
<tr>
<td>Fast R-CNN</td>
<td>VOC07 Trainval + VOC 12 Train</td>
<td>68.4</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>VOC 12 Train (10K)</td>
<td>67.0</td>
</tr>
<tr>
<td>Faster R-CNN</td>
<td>VOC07 Trainval + VOC 12 Train</td>
<td>70.4</td>
</tr>
</tbody>
</table>
The R-CNN family of detectors

<table>
<thead>
<tr>
<th>Detection Model</th>
<th>Mean AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN/10K/VGG</td>
<td>56</td>
</tr>
<tr>
<td>Fast R-CNN/10K/VGG</td>
<td>62</td>
</tr>
<tr>
<td>Fast R-CNN/20K/VGG</td>
<td>66</td>
</tr>
<tr>
<td>Faster R-CNN/20K/VGG</td>
<td>70</td>
</tr>
<tr>
<td>Faster R-CNN/20K/ResNet101</td>
<td>74</td>
</tr>
</tbody>
</table>
Semantic Segmentation
The Task
Evaluation metric

• Pixel classification!
• Accuracy?
  • Heavily unbalanced
  • Common classes are over-emphasized
• Intersection over Union
  • Average across classes and images
• Per-class accuracy
  • Compute accuracy for every class and then average
Things vs Stuff

THINGS
- Person, cat, horse, etc
- Constrained shape
- Individual instances with separate identity
- May need to look at objects

STUFF
- Road, grass, sky etc
- Amorphous, no shape
- No notion of instances
- Can be done at pixel level
- “texture”
Challenges in data collection

• Precise localization is hard to annotate

• Annotating every pixel leads to heavy tails

• Common solution: annotate few classes (often things), mark rest as “Other”

• Common datasets: PASCAL VOC 2012 (~1500 images, 20 categories), COCO (~100k images, 20 categories)
Pre-convnet semantic segmentation

• Things
  • Do object detection, then segment out detected objects

• Stuff
  • ”Texture classification”
  • Compute histograms of filter responses
  • Classify local image patches
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks

Can be considered as a feature vector for a pixel
Semantic segmentation using convolutional networks

Convolve with #classes 1x1 filters

h/4

w/4
Semantic segmentation using convolutional networks

- Pass image through convolution and subsampling layers
- Final convolution with #classes outputs
- Get scores for subsampled image
- Upsample back to original size
Semantic segmentation using convolutional networks
The resolution issue

• Problem: Need fine details!

• Shallower network / earlier layers?
  • Deeper networks work better: more abstract concepts
  • Shallower network => Not very semantic!

• Remove subsampling?
  • Subsampling allows later layers to capture larger and larger patterns
  • Without subsampling => Looks at only a small window!
Solution 1: Image pyramids

Higher resolution  
Less context

Small networks that maintain resolution

Solution 2: Skip connections

Compute class scores at multiple layers, then upsample and add.
Solution 2: Skip connections

Red arrows indicate backpropagation
Skip connections

Skip connections

- Problem: early layers not semantic

Solution 3: Dilation

- Need subsampling to allow convolutional layers to capture large regions with small filters
  - Can we do this without subsampling?
Solution 3: Dilation

• Need subsampling to allow convolutional layers to capture large regions with small filters
  • Can we do this without subsampling?
Solution 3: Dilation

• Need subsampling to allow convolutional layers to capture large regions with small filters
  • Can we do this without subsampling?
Solution 3: Dilation

• Instead of subsampling by factor of 2: dilate by factor of 2

• Dilation can be seen as:
  • Using a much larger filter, but with most entries set to 0
  • Taking a small filter and “exploding”/ “dilating” it

• Not panacea: without subsampling, feature maps are much larger: memory issues
Putting it all together

Other additions

<table>
<thead>
<tr>
<th>Method</th>
<th>mean IoU (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG16 + Skip + Dilation</td>
<td>65.8</td>
</tr>
<tr>
<td>ResNet101</td>
<td>68.7</td>
</tr>
<tr>
<td>ResNet101 + Pyramid</td>
<td>71.3</td>
</tr>
<tr>
<td>ResNet101 + Pyramid + COCO</td>
<td>74.9</td>
</tr>
</tbody>
</table>

Image-to-image translation problems
Image-to-image translation problems

- Segmentation
- Optical flow estimation
- Depth estimation
- Normal estimation
- Boundary detection
- ...

problems