Detection, Segmentation and Fine-grained Localization

Bharath Hariharan, Pablo Arbeláez, Ross Girshick and Jitendra Malik

UC Berkeley
What is image 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*
Goal

A detection system that can describe detected objects in excruciating detail

• Segmentation
• Parts
• Attributes
• 3D models

...
Outline

• Define Simultaneous Detection and Segmentation (SDS) task and benchmark
• SDS by classifying object proposals
• SDS by predicting figure-ground masks
• Part labeling and pose estimation
• Future work and conclusion
Papers


SDS: DEFINING THE TASK AND BENCHMARK
Background: Evaluating object detectors

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

\[
\text{Overlap} = \frac{\text{\# of overlapping pixels}}{\text{\# of all pixels in box}}
\]
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
• If overlap > thresh, correct
• Compute precision-recall (PR) curve
• Compute area under PR curve: Average Precision (AP)

Overlap = \frac{\text{area of intersection}}{\text{area of union}}
Evaluating segments

- Algorithm outputs ranked list of **segments** with category labels
- Compute **region overlap** of each detection with ground truth instances

\[
\text{region} = \frac{\text{overlap}}{\text{union}}
\]
Evaluation metric

- Algorithm outputs ranked list of segments with category labels
- Compute region overlap of each detection with ground truth instances

region = overlap
Evaluation metric

- Algorithm outputs ranked list of segments with category labels
- Compute region overlap of each detection with ground truth instances

\[
\text{region} = \frac{\text{overlap}}{\text{overlap}}
\]
Evaluating segments

- Algorithm outputs ranked list of segments with category labels
- Compute region overlap of each detection with ground truth instances
- If overlap > thresh, correct
- Compute precision-recall (PR) curve
- Compute area under PR curve: Average Precision (APr)

\[
\text{region} = \frac{\text{intersection}}{\text{union}}
\]
Region overlap vs Box overlap

Slide adapted from Philipp Krähenbühl
SDS BY CLASSIFYING BOTTOM-UP CANDIDATES
Background: Bottom-up Object Proposals

- **Motivation:** Reduce search space
- **Aim for recall**
- **Many methods**
  - Multiple segmentations (Selective Search)
  - Combinatorial grouping (MCG)
  - Seed/Graph-cut based (CPMC, GOP)
  - Contour based (Edge Boxes)
Background: CNN

- Neocognitron
  Fukushima, 1980
- Learning Internal Representations by Error Propagation
  Rumelhart, Hinton and Williams, 1986
- Backpropagation applied to handwritten zip code recognition
  Le Cun et al., 1989

- ImageNet Classification with Deep Convolutional Neural Networks
  Krizhevsky, Sutskever and Hinton, 2012

Slide adapted from Ross Girshick
Background: R-CNN

From boxes to segments

Step 1: Generate region proposals

From boxes to segments
Step 2: Score proposals

Box CNN
Region CNN
From boxes to segments
Step 2: Score proposals

Person?
+3.5
+2.6
+0.9
Network training

*Joint task-specific training*

Good region? Yes

Train entire network as one with *region* labels
Network training

**Baseline 1: Separate task specific training**

- Good box? Yes
- Good region? Yes

Train Box CNN using bounding box labels

Train Region CNN using *region* labels
Network training

**Baseline 2:** *Copies of single CNN trained on bounding boxes*

- **Good box? Yes**
- **Box CNN** → **Loss**
- **Copy the weights into Region CNN**

Train Box CNN using bounding box labels
Experiments

• Dataset : PASCAL VOC 2012 / SBD [1]
• Network architecture : [2]

<table>
<thead>
<tr>
<th></th>
<th>AP^r at 0.5</th>
<th>AP^r at 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint</td>
<td>47.7</td>
<td>22.9</td>
</tr>
<tr>
<td>Baseline 1</td>
<td>47.0</td>
<td>21.9</td>
</tr>
<tr>
<td>Baseline 2</td>
<td>42.9</td>
<td>18.0</td>
</tr>
</tbody>
</table>

• Joint, task-specific training works!

Results
Error modes
SDS BY TOP-DOWN FIGURE-GROUND PREDICTION
The need for top-down predictions

• Bottom-up processes make mistakes.
• Some categories have distinctive shapes.
Top-down figure-ground prediction

• Pixel classification
  – For each p in window, does it belong to object?

• Idea: Use features from CNN
CNNs for figure-ground

• Idea: Use features from CNN
• But which layer?
  – Top layers lose localization information
  – Bottom layers are not semantic enough
• Our solution: use all layers!

Figure from: M. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In ECCV 2014.
convolution + pooling

Resize

convolution + pooling

Resize

Resize
Hypercolumns*


Analogy with image pyramids

Hard: large coarse displacements
Easy: small fine deformations

Easy: large coarse displacements
Hard: small fine deformations
Analogy with image pyramids

Hard: large coarse displacements
Easy: small fine deformations

Easy: large coarse displacements
Hard: small fine deformations
Analogy with image pyramids

High resolution “vertical bar” detector

Medium resolution “animal leg” detector

High resolution “horse” detector
Hypercolumns

• Layer outputs are feature maps
• Concatenate to get hypercolumn feature maps
• Feature maps are of coarser resolution
  – Resize (bilinear interpolate) to image resolution
Efficient pixel classification

- Upsampling large feature maps is expensive!
- Linear classification (bilinear interpolation) = bilinear interpolation (linear classification)
- Linear classification = 1x1 convolution
  - extension: use nxn convolution
- Classification = convolve, upsample, sum, sigmoid
convolution + pooling

convolution + pooling

Hypercolumn classifier
Using pixel location
Using pixel location

• Separate classifier for each location?
  – Too expensive
  – Risk of overfitting

• Interpolate into coarse grid of classifiers

\[ f(x) = \alpha f_2(x) + (1 - \alpha) f_1(x) \]
Representation as a neural network
Using top-down predictions

• For refining bottom-up proposals
  – Start from high scoring SDS detections
  – Use hypercolumn features + binary mask to predict figure-ground

• For segmenting bounding box detections
# Refining proposals

<table>
<thead>
<tr>
<th></th>
<th>$\text{AP}^r$ at 0.5</th>
<th>$\text{AP}^r$ at 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>No refinement</td>
<td>47.7</td>
<td>22.8</td>
</tr>
<tr>
<td>Top layer (layer 7)</td>
<td>49.7</td>
<td>25.8</td>
</tr>
<tr>
<td>Layers 7, 4, and 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layers 7 and 2</td>
<td>51.0</td>
<td>31.2</td>
</tr>
<tr>
<td>Layers 4 and 2</td>
<td>50.7</td>
<td>30.8</td>
</tr>
</tbody>
</table>

Refining proposals
Refining proposals: Using multiple layers
Refining proposals: Using multiple layers

Image

Layer 7

Bottom-up candidate

Layers 7, 4 and 2
Refining proposals: Using location

<table>
<thead>
<tr>
<th>Grid size</th>
<th>$\text{AP}^{r}$ at 0.5</th>
<th>$\text{AP}^{r}$ at 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1x1</td>
<td>50.3</td>
<td>28.8</td>
</tr>
<tr>
<td>2x2</td>
<td>51.2</td>
<td>30.2</td>
</tr>
<tr>
<td>5x5</td>
<td>51.3</td>
<td>31.8</td>
</tr>
<tr>
<td>10x10</td>
<td>51.2</td>
<td>31.6</td>
</tr>
</tbody>
</table>
Refining proposals: Using location

1 x 1

5 x 5
Refining proposals: Finetuning and bbox regression

<table>
<thead>
<tr>
<th></th>
<th>APR at 0.5</th>
<th>APR at 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypercolumn</td>
<td>51.2</td>
<td>31.6</td>
</tr>
<tr>
<td>+Bbox Regression</td>
<td>51.9</td>
<td>32.4</td>
</tr>
<tr>
<td>+Bbox Regression+FT</td>
<td><strong>52.8</strong></td>
<td><strong>33.7</strong></td>
</tr>
</tbody>
</table>
Segmenting bbox detections
## Segmenting bbox detections

<table>
<thead>
<tr>
<th>Network</th>
<th>APr at 0.5</th>
<th>APr at 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify segments + Refine</td>
<td>T-net[1]</td>
<td>51.9</td>
</tr>
</tbody>
</table>

Segment + Rescore
## Segmenting bbox detections

<table>
<thead>
<tr>
<th></th>
<th>Network</th>
<th>APr at 0.5</th>
<th>APr at 0.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classify segments + Refine</td>
<td>T-net[1]</td>
<td>51.9</td>
<td>32.4</td>
</tr>
<tr>
<td>Segment bbox detections</td>
<td>T-net</td>
<td>49.1</td>
<td>29.1</td>
</tr>
<tr>
<td>Segment bbox detections</td>
<td>O-net[2]</td>
<td>56.5</td>
<td>37.0</td>
</tr>
</tbody>
</table>

Qualitative results
Qualitative results
Error modes

Multiple objects

Non-prototypical poses

Occlusion
Summary of SDS

APr at 0.7

- Proposal classification
- Proposal classification + top layer refinement
- Proposal classification + hypercolumn refinement
- Detection + f/g prediction + rescoring (O-net)
Part Labeling

• Same (hypercolumn) features, different labels!
Part Labeling - Experiments

• Dataset: PASCAL Parts [1]
• Evaluation: Detection is correct if \( \frac{\text{#(correctly labeled pixels)}}{\text{union}} > \text{threshold} \)

<table>
<thead>
<tr>
<th></th>
<th>Bird</th>
<th>Cat</th>
<th>Cow</th>
<th>Dog</th>
<th>Horse</th>
<th>Person</th>
<th>Sheep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer 7</td>
<td>15.4</td>
<td>19.2</td>
<td>14.5</td>
<td>8.5</td>
<td>16.6</td>
<td>21.9</td>
<td>38.9</td>
</tr>
<tr>
<td>Layers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7, 4, 2</td>
<td>14.2</td>
<td>30.3</td>
<td>21.5</td>
<td>14.2</td>
<td>27.8</td>
<td>28.5</td>
<td>44.9</td>
</tr>
</tbody>
</table>

Error modes

Disjointed parts

Misclassification

Wrong figure/ground
Conclusion

• A detection system that can
  – Provide pixel accurate segmentations
  – Provide part labelings and pose estimates

• A general framework for fine-grained localization using CNNs.