CNNs for dense image labeling

- Image classification
- Object detection
- Semantic segmentation
- Instance segmentation
- Panoptic segmentation
Outline

• Early “hacks”
  • Hypercolumns
  • Zoom-out features
  • Fully convolutional networks

• Deep network operations for dense prediction
  • Transposed convolutions
  • Unpooling
  • Dilated convolutions

• Instance segmentation
  • Mask R-CNN

• Other dense prediction problems
Early “hacks”

- Do dense prediction as a post-process on top of an image classification CNN

**Have:** feature maps from image classification network

**Want:** pixel-wise predictions
Hypercolumns

• Idea: to obtain a feature representation for an individual pixel, upsample all feature maps to original image resolution and concatenate values from feature maps “above” that pixel

Zoom-out features

M. Mostajabi, P. Yadollahpour and G. Shakhnarovich, Feedforward semantic segmentation with zoom-out features, CVPR 2015
Zoom-out features: Example results
Zoom-out features: Evaluation

• **Metric: mean IoU**
  • Intersection over union of predicted and ground truth pixels for each class, averaged over classes

<table>
<thead>
<tr>
<th>Method</th>
<th>VOC2010</th>
<th>VOC2011</th>
<th>VOC2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>zoom-out (ours)</td>
<td>69.9</td>
<td>69.4</td>
<td>69.6</td>
</tr>
<tr>
<td>FCN-8s [26]</td>
<td>–</td>
<td>62.7</td>
<td>62.2</td>
</tr>
<tr>
<td>DivMbest+convnet [8]</td>
<td>–</td>
<td>–</td>
<td>52.2</td>
</tr>
<tr>
<td>SDS [15]</td>
<td>–</td>
<td>52.6</td>
<td>51.6</td>
</tr>
<tr>
<td>DivMbest+rerank [39]</td>
<td>–</td>
<td>–</td>
<td>48.1</td>
</tr>
<tr>
<td>O2P [4]</td>
<td>–</td>
<td>47.6</td>
<td>47.8</td>
</tr>
<tr>
<td>Regions &amp; parts[2]</td>
<td>–</td>
<td>40.8</td>
<td>–</td>
</tr>
<tr>
<td>D-sampling [27]</td>
<td>33.5</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>Harmony potentials [3]</td>
<td>40.1</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class</th>
<th>mean</th>
<th>acc</th>
<th>91.9</th>
<th>85.6</th>
<th>37.3</th>
<th>83.2</th>
<th>62.5</th>
<th>66</th>
<th>85.1</th>
<th>80.7</th>
<th>84.9</th>
<th>27.2</th>
<th>73.3</th>
<th>57.5</th>
<th>78.1</th>
<th>79.2</th>
<th>81.1</th>
<th>77.1</th>
<th>53.6</th>
<th>74</th>
<th>49.2</th>
<th>71.7</th>
<th>63.3</th>
</tr>
</thead>
</table>
Fully convolutional networks

- Design a network with only convolutional layers, make predictions for all pixels at once

Fully convolutional networks

- Design a network with only convolutional layers, make predictions for all pixels at once
- Can the network operate at full image resolution?

Source: Stanford CS231n
Fully convolutional networks

- Design a network with only convolutional layers, make predictions for all pixels at once
- Can the network operate at full image resolution?
- Practical solution: first downsample, then upsample

Source: Stanford CS231n
Fully convolutional networks (FCN)

Comparison on a subset of PASCAL 2011 validation data:

<table>
<thead>
<tr>
<th>Model</th>
<th>Pixel acc.</th>
<th>Mean acc.</th>
<th>Mean IU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN-32s-fixed</td>
<td>83.0</td>
<td>59.7</td>
<td>45.4</td>
</tr>
<tr>
<td>FCN-32s</td>
<td>89.1</td>
<td>73.3</td>
<td>59.4</td>
</tr>
<tr>
<td>FCN-16s</td>
<td>90.0</td>
<td>75.7</td>
<td>62.4</td>
</tr>
<tr>
<td>FCN-8s</td>
<td><strong>90.3</strong></td>
<td><strong>75.9</strong></td>
<td><strong>62.7</strong></td>
</tr>
</tbody>
</table>

Outline

• Early “hacks”
  • Hypercolumns
  • Zoom-out features
  • Fully convolutional networks

• Deep network operations for dense prediction
  • Transposed convolutions
  • Unpooling
  • Dilated convolutions
Transposed convolution

- Regular convolution (stride 1, pad 0)

\[
\begin{bmatrix}
  x_{11} & x_{12} & x_{13} & x_{14} \\
  x_{21} & x_{22} & x_{23} & x_{24} \\
  x_{31} & x_{32} & x_{33} & x_{34} \\
  x_{41} & x_{42} & x_{43} & x_{44}
\end{bmatrix} \ast \begin{bmatrix}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33} \\
  w_{41}
\end{bmatrix} = \begin{bmatrix}
  z_{11} & z_{12} \\
  z_{21} & z_{22}
\end{bmatrix}
\]

- Matrix-vector form:

\[
\begin{bmatrix}
  w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 & 0 & 0 & 0 & 0 \\
  0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 \\
  0 & 0 & 0 & 0 & 0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33}
\end{bmatrix}\begin{bmatrix}
  x_{11} \\
  x_{12} \\
  x_{13} \\
  x_{14} \\
  \vdots \\
  x_{44}
\end{bmatrix} = \begin{bmatrix}
  z_{11} \\
  z_{12} \\
  z_{21} \\
  z_{22}
\end{bmatrix}
\]

4x4 input, 2x2 output
Transposed convolution

Let’s transpose the filter matrix:

\[
\begin{bmatrix}
  z_{11} & z_{12} \\
  z_{21} & z_{22}
\end{bmatrix}
\star^T
\begin{bmatrix}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33}
\end{bmatrix}
= \\
\begin{bmatrix}
  x_{11} & x_{12} & x_{13} & x_{14} \\
  x_{21} & x_{22} & x_{23} & x_{24} \\
  x_{31} & x_{32} & x_{33} & x_{34} \\
  x_{41} & x_{42} & x_{43} & x_{44}
\end{bmatrix}
\]

2x2 input, 4x4 output

*Not* an inverse of the original convolution operation, simply reverses dimension change!

(sometimes called *deconvolution* instead of transposed convolution, but this is misleading)
Transposed convolution

\[
\begin{bmatrix}
z_{11} & z_{12} \\
\hline
z_{21} & z_{22}
\end{bmatrix}
\ast^T
\begin{bmatrix}
w_{11} & w_{12} & w_{13} \\
\hline
w_{21} & w_{22} & w_{23}
\end{bmatrix}
= \begin{bmatrix}
x_{11} & x_{12} & x_{13} & x_{14} \\
\hline
x_{21} & x_{22} & x_{23} & x_{24}
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix}
w_{11} & 0 & 0 & 0 \\
\hline
w_{12} & w_{11} & 0 & 0 \\
w_{13} & w_{12} & 0 & 0 \\
0 & w_{13} & 0 & 0 \\
w_{21} & 0 & w_{11} & 0 \\
w_{22} & w_{21} & w_{12} & w_{11} \\
w_{23} & w_{22} & w_{13} & w_{12} \\
0 & w_{23} & 0 & w_{13} \\
w_{31} & 0 & w_{21} & 0 \\
w_{32} & w_{31} & w_{22} & w_{21} \\
w_{33} & w_{32} & w_{23} & w_{22} \\
0 & w_{33} & 0 & w_{23} \\
0 & 0 & w_{31} & 0 \\
0 & 0 & w_{32} & w_{31} \\
0 & 0 & w_{33} & w_{32} \\
0 & 0 & 0 & w_{33}
\end{bmatrix}
\end{bmatrix}
\begin{bmatrix}
z_{11} \\
\hline
z_{12} \\
z_{21} \\
z_{22}
\end{bmatrix}
= \begin{bmatrix}
x_{11} = w_{11}z_{11}
\end{bmatrix}
\]
Transposed convolution

Convolve input with flipped filter

\[ x_{12} = w_{12}z_{11} + w_{11}z_{12} \]
Transposed convolution

\[ \begin{bmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{bmatrix} \ast^T \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \\ x_{31} & x_{32} & x_{33} & x_{34} \\ x_{41} & x_{42} & x_{43} & x_{44} \end{bmatrix} \]

Convolve input with \textit{flipped} filter

\[ x_{13} = w_{13}z_{11} + w_{12}z_{12} \]
Transposed convolution

Convolve input with flipped filter

$x_{14} = w_{13} z_{12}$
Transposed convolution

Convolve input with flipped filter

\[
x_{21} = w_{21}z_{11} + w_{11}z_{21}
\]
Transposed convolution

Convolve input with flipped filter

\[
\begin{pmatrix}
    x_{11} & x_{12} & x_{13} & x_{14} \\
    x_{21} & x_{22} & x_{23} & x_{24} \\
    x_{31} & x_{32} & x_{33} & x_{34} \\
    x_{41} & x_{42} & x_{43} & x_{44}
\end{pmatrix}
\]

\[
x_{22} = w_{22}z_{11} + w_{21}z_{12} + w_{12}z_{21} + w_{11}z_{22}
\]
Transposed convolution

Convolve input with flipped filter

\[ x_{23} = w_{23}z_{11} + w_{22}z_{12} + w_{13}z_{21} + w_{12}z_{22} \]
Transposed convolution

\[
\begin{bmatrix}
z_{11} & z_{12} \\
z_{21} & z_{22}
\end{bmatrix}
\ast^T
\begin{bmatrix}
w_{11} & w_{12} & w_{13} \\
w_{21} & w_{22} & w_{23} \\
w_{31} & w_{32} & w_{33}
\end{bmatrix}
= 
\begin{bmatrix}
x_{11} & x_{12} & x_{13} & x_{14} \\
x_{21} & x_{22} & x_{23} & x_{24} \\
x_{31} & x_{32} & x_{33} & x_{34} \\
x_{41} & x_{42} & x_{43} & x_{44}
\end{bmatrix}
\]

\[
\begin{bmatrix}
w_{11} & 0 & 0 & 0 \\
w_{12} & w_{11} & 0 & 0 \\
w_{13} & w_{12} & 0 & 0 \\
0 & w_{13} & 0 & 0 \\
w_{21} & 0 & w_{11} & 0 \\
w_{22} & w_{21} & w_{12} & w_{11} \\
w_{23} & w_{22} & w_{13} & w_{12} \\
0 & w_{23} & 0 & w_{13} \\
w_{31} & 0 & w_{21} & 0 \\
w_{32} & w_{31} & w_{22} & w_{21} \\
w_{33} & w_{32} & w_{23} & w_{22} \\
0 & w_{33} & 0 & w_{23} \\
0 & 0 & w_{31} & 0 \\
0 & 0 & w_{32} & w_{31} \\
0 & 0 & w_{33} & w_{32} \\
0 & 0 & 0 & w_{33}
\end{bmatrix}
= 
\begin{bmatrix}
z_{11} \\
z_{12} \\
z_{21} \\
z_{22}
\end{bmatrix}
\]
Transposed convolution

Alternate view
• Place copies of the filter on the output, weighted by entries of the input
Transposed convolution

Alternate view
- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap
Transposed convolution

Alternate view

- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap
Transposed convolution

Alternate view

- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap
Transposed convolution

• 1D example

Animation: [Link](https://distill.pub/2016/deconv-checkerboard/)
Transposed convolution

- 1D example

Animation: [Link to Animation](https://distill.pub/2016/deconv-checkerboard/)
Transposed convolution

- 1D example

![Diagram of transposed convolution]

input \(x_1 \ x_2 \ x_3 \ x_4 \ x_5 \ x_6 \ x_7 \ x_8\)

output \(w_3 x_1 + w_2 x_2 + w_1 x_3\)

filter: \([w_1 \ w_2 \ w_3]\)

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Transposed convolution

- 1D example

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- *Backwards-strided convolution*: to increase resolution, use *output stride* $> 1$

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling by transposed convolution

- **Backwards-strided convolution**: to increase resolution, use output stride $>1$

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- *Backwards-strided convolution*: to increase resolution, use output stride > 1

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- **Backwards-strided convolution**: to increase resolution, use output stride > 1

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- *Backwards-strided convolution*: to increase resolution, use *output stride* > 1

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling by transposed convolution

- **Backwards-strided convolution**: to increase resolution, use *output stride* > 1
  - For stride 2, dilate the input by inserting rows and columns of zeros between adjacent entries, convolve with flipped filter
  - Sometimes called convolution with *fractional input stride* 1/2

Q: What 3x3 filter would correspond to bilinear upsampling?

V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, arXiv 2018
Outline

• Early “hacks”
  • Hypercolumns
  • Zoom-out features
  • Fully convolutional networks

• Deep network operations for dense prediction
  • Transposed convolutions
  • Unpooling
Upsampling by unpooling

- Alternative to transposed convolution: max unpooling

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>6</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

Max pooling

<table>
<thead>
<tr>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Remember pooling indices (which element was max)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>6</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>0</td>
<td>8</td>
</tr>
</tbody>
</table>

Max unpooling

Output is sparse, so unpooling is typically followed by a transposed convolution layer
DeconvNet

DeconvNet

H. Noh, S. Hong, and B. Han, Learning Deconvolution Network for Semantic Segmentation, ICCV 2015
### DeconvNet results

<table>
<thead>
<tr>
<th>PASCAL VOC 2012</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hypercolumns</td>
<td>59.2</td>
</tr>
<tr>
<td>ZoomOut</td>
<td>64.4</td>
</tr>
<tr>
<td>FCN-8</td>
<td>62.2</td>
</tr>
<tr>
<td>DeconvNet</td>
<td>69.6</td>
</tr>
<tr>
<td>Ensemble of DeconvNet and FCN</td>
<td>71.7</td>
</tr>
</tbody>
</table>
Similar architecture: SegNet


Drop the FC layers, get better results
U-Net

• Like FCN, fuse upsampled higher-level feature maps with higher-res, lower-level feature maps
• Unlike FCN, fuse by concatenation, predict at the end

Summary of upsampling architectures
Recall: Feature pyramid networks

- Improve predictive power of lower-level feature maps by adding contextual information from higher-level feature maps.
- Predict different sizes of bounding boxes from different levels of the pyramid (but share parameters of predictors).

Outline

• Early “hacks”
  • Hypercolumns
  • Zoom-out features
  • Fully convolutional networks

• Deep network operations for dense prediction
  • Transposed convolutions
  • Unpooling
  • Dilated convolutions
Dilated convolutions

- Idea: instead of reducing spatial resolution of feature maps, use a large sparse filter
  - Also known as à trous convolution
Dilated convolutions

• Idea: instead of reducing spatial resolution of feature maps, use a large sparse filter

Like 2x downsampling followed by 3x3 convolution followed by 2x upsampling

V. Dumoulin and F. Visin, A guide to convolution arithmetic for deep learning, arXiv 2018
Dilated convolutions

Dilated convolutions

- Can be used in FCN to remove downsampling: change stride of max pooling layer from 2 to 1, dilate subsequent convolutions by factor of 2 (in theory, can be done without re-training any parameters)

Dilated convolutions

- Can increase receptive field size exponentially with a linear growth in the number of parameters

F. Yu and V. Koltun, Multi-scale context aggregation by dilated convolutions, ICLR 2016
Dilated convolutions

- Context module with dilation
  - Returns same number of feature maps at the same resolution as the input, so can be plugged in to replace components of existing dense prediction architectures
  - Requires identity initialization

<table>
<thead>
<tr>
<th>Layer</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>$3 \times 3$</td>
<td>$3 \times 3$</td>
<td>$3 \times 3$</td>
<td>$3 \times 3$</td>
<td>$3 \times 3$</td>
<td>$3 \times 3$</td>
<td>$3 \times 3$</td>
<td>$1 \times 1$</td>
</tr>
<tr>
<td>Dilation</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>8</td>
<td>16</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Truncation</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Receptive field</td>
<td>$3 \times 3$</td>
<td>$5 \times 5$</td>
<td>$9 \times 9$</td>
<td>$17 \times 17$</td>
<td>$33 \times 33$</td>
<td>$65 \times 65$</td>
<td>$67 \times 67$</td>
<td>$67 \times 67$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
</tr>
<tr>
<td>Large</td>
</tr>
</tbody>
</table>

F. Yu and V. Koltun, [Multi-scale context aggregation by dilated convolutions](http://iclr.cc), ICLR 2016
Dilated convolutions: Evaluation

Results on VOC 2012

<table>
<thead>
<tr>
<th></th>
<th>aero</th>
<th>bike</th>
<th>bird</th>
<th>boat</th>
<th>bottle</th>
<th>bus</th>
<th>car</th>
<th>cat</th>
<th>chair</th>
<th>cow</th>
<th>table</th>
<th>dog</th>
<th>horse</th>
<th>mbike</th>
<th>person</th>
<th>plant</th>
<th>sheep</th>
<th>sofa</th>
<th>train</th>
<th>tv</th>
<th>mean IoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Front end</td>
<td>86.3</td>
<td>38.2</td>
<td>76.8</td>
<td>66.8</td>
<td>63.2</td>
<td>87.3</td>
<td>78.7</td>
<td>82</td>
<td>33.7</td>
<td>76.7</td>
<td>53.5</td>
<td>73.7</td>
<td>76</td>
<td>83</td>
<td>51.9</td>
<td>77.8</td>
<td>44</td>
<td>79.9</td>
<td>66.3</td>
<td>69.8</td>
<td></td>
</tr>
<tr>
<td>Front + Basic</td>
<td>86.4</td>
<td>37.6</td>
<td>78.5</td>
<td>66.3</td>
<td>64.1</td>
<td>89.9</td>
<td>79.9</td>
<td>84.9</td>
<td>36.1</td>
<td>79.4</td>
<td>55.8</td>
<td>77.6</td>
<td>81.6</td>
<td>79</td>
<td>83.1</td>
<td>51.2</td>
<td>81.3</td>
<td>43.7</td>
<td>82.3</td>
<td>65.7</td>
<td></td>
</tr>
<tr>
<td>Front + Large</td>
<td>87.3</td>
<td>39.2</td>
<td>80.3</td>
<td>65.6</td>
<td>66.4</td>
<td>90.2</td>
<td>82.6</td>
<td>85.8</td>
<td>34.8</td>
<td>81.9</td>
<td>51.7</td>
<td>79</td>
<td>84.1</td>
<td>80.9</td>
<td>83.2</td>
<td>51.2</td>
<td>83.2</td>
<td>44.7</td>
<td>83.4</td>
<td>65.6</td>
<td></td>
</tr>
</tbody>
</table>

*Front end: re-implementation of FCN-8 with last two pooling layers dropped (5% better than original FCN-8)

F. Yu and V. Koltun, Multi-scale context aggregation by dilated convolutions, ICLR 2016
Dilated convolutions: Evaluation

(a) Image  (b) Front end  (c) + Context  (d) + CRF-RNN  (e) Ground truth
Semantic segmentation: Outline

• Early “hacks”
  • Hypercolumns
  • Zoom-out features
  • Fully convolutional networks

• Deep network operations for dense prediction
  • Transposed convolutions
  • Unpooling
  • Dilated convolutions

• Instance segmentation
  • Mask R-CNN
Instance segmentation

Object Detection  Semantic Segmentation  Instance Segmentation

Source: Kaiming He
Mask R-CNN

- Mask R-CNN = Faster R-CNN + FCN on RoIs

K. He, G. Gkioxari, P. Dollar, and R. Girshick, [Mask R-CNN](https://arxiv.org/abs/1703.06870), ICCV 2017 (Best Paper Award)
RoIAlign vs. RoIPool

- RoIPool: nearest neighbor quantization

K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)
RoIAlign vs. RoIPool

- RoIPool: nearest neighbor quantization
- RoIAlign: bilinear interpolation

Mask R-CNN

- From RoIAlign features, predict class label, bounding box, and segmentation mask

Classification/regression head from an established object detector (e.g., FPN)

Separately predict binary mask for each class with per-pixel sigmoids, use average binary cross-entropy loss

K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)
Mask R-CNN

K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)
Example results
Example results
### Instance segmentation results on COCO

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>AP</th>
<th>AP$_{50}$</th>
<th>AP$_{75}$</th>
<th>AP$_{S}$</th>
<th>AP$_{M}$</th>
<th>AP$_{L}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNC [10]</td>
<td>ResNet-101-C4</td>
<td>24.6</td>
<td>44.3</td>
<td>24.8</td>
<td>4.7</td>
<td>25.9</td>
<td>43.6</td>
</tr>
<tr>
<td>FCIS [26] +OHEM</td>
<td>ResNet-101-C5-dilated</td>
<td>29.2</td>
<td>49.5</td>
<td>-</td>
<td>7.1</td>
<td>31.3</td>
<td>50.0</td>
</tr>
<tr>
<td>FCIS+++ [26] +OHEM</td>
<td>ResNet-101-C5-dilated</td>
<td>33.6</td>
<td>54.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNet-101-C4</td>
<td>33.1</td>
<td>54.9</td>
<td>34.8</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNet-101-FPN</td>
<td>35.7</td>
<td>58.0</td>
<td>37.8</td>
<td>12.1</td>
<td>35.6</td>
<td>51.1</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNeXt-101-FPN</td>
<td>37.1</td>
<td>60.0</td>
<td>39.4</td>
<td>16.9</td>
<td>39.9</td>
<td>53.5</td>
</tr>
</tbody>
</table>

*AP at different IoU thresholds*  
*AP for different size instances*

K. He, G. Gkioxari, P. Dollar, and R. Girshick, Mask R-CNN, ICCV 2017 (Best Paper Award)
Keypoint prediction

- Given $K$ keypoints, train model to predict $K \times m \times m$ one-hot maps with cross-entropy losses over $m^2$ outputs
Outline

• Early “hacks”
  • Hypercolumns
  • Zoom-out features
  • Fully convolutional networks

• Deep network operations for dense prediction
  • Transposed convolutions
  • Unpooling
  • Dilated convolutions

• Instance segmentation
  • Mask R-CNN

• Other dense prediction problems
Recently proposed task: Panoptic segmentation

A. Kirillov et al. Panoptic segmentation. CVPR 2019
Panoptic feature pyramid networks

A. Kirillov et al. Panoptic feature pyramid networks. CVPR 2019
Panoptic feature pyramid networks

Figure 2: Panoptic FPN results on COCO (top) and Cityscapes (bottom) using a single ResNet-101-FPN network.

A. Kirillov et al. Panoptic feature pyramid networks, CVPR 2019
Another recent task: Amodal instance segmentation

K. Li and J. Malik. Amodal instance segmentation, ECCV 2016
Even more dense prediction problems

- Depth estimation
- Surface normal estimation
- Colorization
- ....
Depth and normal estimation

Depth and normal estimation