CNNs for dense image labeling

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

• Early “hacks”
  • Hypercolumns
  • Zoom-out features

• Fully convolutional networks
  • Learned upsampling architectures
  • 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>bg</th>
<th>plane</th>
<th>train</th>
</tr>
</thead>
<tbody>
<tr>
<td>acc</td>
<td>69.6</td>
<td>91.9</td>
<td>85.6</td>
<td>37.3</td>
</tr>
</tbody>
</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.
- Ideally, we want convolutions at full image resolution, but implementing that naively is too expensive.

Source: Stanford CS231n
Fully convolutional networks

- Design a network with only convolutional layers, make predictions for all pixels at once
- Ideally, we want convolutions at full image resolution, but implementing that naively is too expensive
  - Solution: first downsample, then upsample

Source: Stanford CS231n
Upsampling in a deep network

- Regular convolution (stride 1, pad 0)

\[
\begin{array}{cccc}
  x_{11} & x_{12} & x_{13} & x_{14} \\
  x_{21} & x_{22} & x_{23} & x_{24} \\
  x_{31} & x_{32} & x_{33} & x_{34} \\
  x_{41} & x_{42} & x_{43} & x_{44} \\
\end{array}
\] \quad *
\begin{array}{ccc}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33} \\
\end{array} =
\begin{array}{cc}
  z_{11} & z_{12} \\
  z_{21} & z_{22} \\
\end{array}
\]

- Matrix-vector form:

\[
\begin{pmatrix}
w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 & 0 & 0 & 0 & 0 \\
0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 \\
0 & 0 & 0 & 0 & 0 & w_{11} & w_{12} & w_{13} & 0 & w_{21} & w_{22} & w_{23} & 0 & w_{31} & w_{32} & w_{33} & 0 \\
\end{pmatrix}
\begin{pmatrix}
x_{11} \\
x_{12} \\
x_{13} \\
x_{14} \\
x_{41} \\
x_{42} \\
x_{43} \\
x_{44} \\
\end{pmatrix} =
\begin{pmatrix}
z_{11} \\
z_{12} \\
z_{21} \\
z_{22} \\
\end{pmatrix}
\]

4x4 input, 2x2 output
Upsampling in a deep network

• Transposed convolution

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

2x2 input, 4x4 output

\( \text{Not} \) an inverse of the original convolution operation, simply reverses dimension change!
Upsampling in a deep network

• Transposed convolution

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

\[
x_{11} = w_{11} z_{11}
\]
Upsampling in a deep network

- Transposed convolution

\[
\begin{pmatrix}
  z_{11} & z_{12} \\
  z_{21} & z_{22}
\end{pmatrix}
\ast T
\begin{pmatrix}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33}
\end{pmatrix}
\]

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

Convolve input with flipped filter

\[
x_{12} = w_{12}z_{11} + w_{11}z_{12}
\]
Upsampling in a deep network

- Transposed convolution

Convolve input with flipped filter

\[
\begin{bmatrix}
  z_{11} & z_{12} \\
  z_{21} & z_{22}
\end{bmatrix} * T
\]

\[
\begin{bmatrix}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33}
\end{bmatrix}
\]

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

\[
x_{13} = w_{13}z_{11} + w_{12}z_{12}
\]
Upsampling in a deep network

- Transposed convolution

Convolve input with flipped filter
Upsampling in a deep network

- Transposed convolution

Convolve input with flipped filter

\[
x_{21} = w_{21}z_{11} + w_{11}z_{21}
\]
Upsampling in a deep network

- Transposed convolution

\[
\begin{bmatrix}
  w_{11} & w_{12} & w_{13} \\
  w_{21} & w_{22} & w_{23} \\
  w_{31} & w_{32} & w_{33}
\end{bmatrix} \ast T
\]

Convolve input with \textit{flipped} filter

\[
x_{22} = w_{22}z_{11} + w_{21}z_{12} + w_{12}z_{21} + w_{11}z_{22}
\]
Upsampling in a deep network

- Transposed convolution

Convolve input with flipped filter

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

\[x_{23} = w_{23}z_{11} + w_{22}z_{12} + w_{13}z_{21} + w_{12}z_{22}\]
Upsampling in a deep network

- Transposed convolution

V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, arXiv 2018
Recall: Backward pass for conv layer

\[
\frac{\partial e}{\partial x_{kl}} = \sum_{i,j} \frac{\partial e}{\partial z_{k+i, l+j}} \frac{\partial z_{k+i, l+j}}{\partial x_{kl}} = \sum_{i,j} \frac{\partial e}{\partial z_{k+i, l+j}} w_{-i, -j}
\]

- Transposed convolution is the same operation as backwards pass for regular convolution
Upsampling in a deep network

- Transposed convolution

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

- Transposed convolution

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

Alternate view:
- Place copies of the filter on the output, weighted by entries of the input
Upsampling in a deep network

• 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
Upsampling in a deep network

- Transposed convolution

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

Alternate view:
- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap
Upsampling in a deep network

- Transposed convolution

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

Alternate view:
- Place copies of the filter on the output, weighted by entries of the input
- Sum where copies of the filter overlap
Upsampling in a deep network

- 1D example

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling in a deep network

• 1D example

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling in a deep network

• 1D example

input

output

\[ w_3 x_1 + w_2 x_2 + w_1 x_3 \]
Upsampling in a deep network

- 1D example

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling in a deep network

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

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling in a deep network

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

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling in a deep network

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

Animation: [https://distill.pub/2016/deconv-checkerboard/](https://distill.pub/2016/deconv-checkerboard/)
Upsampling in a deep network

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

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling in a deep network

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

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling in a deep network

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

Animation: https://distill.pub/2016/deconv-checkerboard/
Upsampling in a deep network

- **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?

\[
\begin{bmatrix}
\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\
\frac{1}{2} & 1 & \frac{1}{2} \\
\frac{1}{4} & \frac{1}{2} & \frac{1}{4}
\end{bmatrix}
\]

V. Dumoulin and F. Visin, *A guide to convolution arithmetic for deep learning*, arXiv 2018
Fully convolutional networks (FCN)

Fully convolutional networks (FCN)

- Predictions by 1x1 conv layers, bilinear upsampling
- Predictions by 1x1 conv layers, learned 2x upsampling, fusion by summing

Upsampling in a deep network

- Alternative to transposed convolution: max unpooling

Remember pooling indices (which element was max)

Output is sparse, so need to follow this with a transposed convolution layer

(sometimes called deconvolution instead of transposed convolution, but this is not accurate)
DeconvNet

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

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

O. Ronneberger, P. Fischer, T. Brox U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015
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)

Summary of upsampling architectures

Figure source
Dilated convolutions

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

Dilation factor 1

Dilation factor 2

Dilation factor 3
Dilated convolutions

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

![Diagram of dilated convolution]

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

• Use in FCN to remove downsampling: change stride of max pooling layer from 2 to 1, dilate subsequent convolutions by factor of 2 (possibly 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

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

<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>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>3×3</td>
<td>1×1</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>3×3</td>
<td>5×5</td>
<td>9×9</td>
<td>17×17</td>
<td>33×33</td>
<td>65×65</td>
<td>67×67</td>
<td>67×67</td>
</tr>
</tbody>
</table>

Output channels

<table>
<thead>
<tr>
<th>Basic</th>
<th>Large</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>2C</td>
</tr>
<tr>
<td>C</td>
<td>2C</td>
</tr>
<tr>
<td>C</td>
<td>4C</td>
</tr>
<tr>
<td>C</td>
<td>8C</td>
</tr>
<tr>
<td>C</td>
<td>16C</td>
</tr>
<tr>
<td>C</td>
<td>32C</td>
</tr>
<tr>
<td>C</td>
<td>32C</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
</tr>
</tbody>
</table>
### 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>51.2</td>
<td>81.3</td>
<td>43.7</td>
<td>82.3</td>
<td>65.7</td>
<td>71.3</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>83.4</td>
<td>44.7</td>
<td>83.4</td>
<td>65.6</td>
<td>72.1</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)*
Dilated convolutions: Evaluation
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, 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

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></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>12.1</td>
<td>35.6</td>
<td>51.1</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>15.5</td>
<td>38.1</td>
<td>52.4</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>ResNeXt-101-FPN</td>
<td><strong>37.1</strong></td>
<td><strong>60.0</strong></td>
<td><strong>39.4</strong></td>
<td><strong>16.9</strong></td>
<td><strong>39.9</strong></td>
<td><strong>53.5</strong></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 $m \times m$ one-hot maps
Other dense prediction tasks

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

D. Eigen and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, ICCV 2015
Depth and normal estimation

D. Eigen and R. Fergus, Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture, ICCV 2015
Colorization

R. Zhang, P. Isola, and A. Efros, Colorful Image Colorization, ECCV 2016

<table>
<thead>
<tr>
<th>Input</th>
<th>Boundaries</th>
<th>Saliency</th>
<th>Normals</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detection</th>
<th>Semantic Boundaries &amp; Segmentation</th>
<th>Human Parts</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Figure 2: UberNet architecture: an image pyramid is formed by successive down-sampling operations, and each image is processed by a CNN with tied weights; the responses of the network at consecutive layers (C_i) are processed with Batch Normalization (B_i) and then fed to task-specific skip layers (E_i); these are combined across network layers (F_i) and resolutions (S_i) and trained using task-specific loss functions (L_i), while the whole architecture is jointly trained end-to-end. For simplicity we omit the interpolation and detection layers mentioned in the text.