Semantic segmentation
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks
Semantic segmentation using convolutional networks

Convolve with #classes 1x1 filters

h/4
w/4

#classes
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
Transfer learning for semantic segmentation

Figure 2. Transforming fully connected layers into convolution layers enables a classification net to output a heatmap. Adding layers and a spatial loss (as in Figure 1) produces an efficient machine for end-to-end dense learning.

Semantic segmentation using convolutional networks
The resolution issue

• Problem: Need fine details!
• Shallower network / earlier layers?
  • Not very semantic!
• Remove subsampling?
  • Looks at only a small window!
Solution 1: Image pyramids

Solution 2: Skip connections
Solution 2: Skip connections
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
Solution 4: Conditional random fields

\[ P(y|x) = \frac{1}{Z} e^{-E(y,x)} \]

\[ y^* = \arg \max_y P(y|x) \]
\[ = \arg \min_y E(y, x) \]

\[ E(y, x) = \sum_i E_{data}(y_i, x) + \sum_{i,j \in N} E_{smooth}(y_i, y_j, x) \]
Solution 4: Conditional Random Fields

- Idea: take convolutional network prediction and sharpen using classic techniques
- Conditional Random Field

\[ y^* = \arg \min_y \sum_i E_{data}(y_i, x) + \sum_{i,j \in N} E_{smooth}(y_i, y_j, x) \]

\[ E_{smooth}(y_i, y_j, x) = \mu(y_i, y_j) w_{ij}(x) \]

- Label compatibility
- Pixel similarity
Inference in CRFs

• Problem: combinatorial optimization
• Variational methods: Approximate complex distribution $p(y)$ with simple distribution $q(y)$
• Mean-field approximation: $q(y)$ is independent distribution for each pixel:

$$q(y) = \prod_{i} q_i(y_i)$$

• If N pixels and K classes, basically N K-dimensional vectors
Mean field inference

• If we can find best q, solution is highest probability output for each pixel

• Try to match p with q by minimizing Kulback-Leibler Divergence

\[ KL(q||p) = \sum_y q(y) \log p(y) - \sum_y q(y) \log q(y) \]

• Iterative process: in each iteration, do coordinate ascent on one q(y_i)
Mean field inference

- Coordinate descent on $q_i(y_i)$
- At each step, keep other pixels fixed and update one
- Each step (approximately):
  - Take current $q_j(y_j)$ on all $j \neq i$
  - Use this to compute $p(y_i|y_{-i})$ where $y_{-i} = \{y_j: j \neq i\}$
  - Set $q_i$ to this

$$ q_i \propto \mathbb{E}_{q_{-i}} [\log p(y_i|y_{-i})] $$
Fully Connected CRFs

- Typically, only adjacent pixels connected
  - Fewer connections => Easier to optimize
- Dense connectivity: every pixel connected to everything else
- Intractable to optimize except if pairwise potential takes specific form

\[
E_{\text{smooth}}(y_i, y_j, x) = \mu(y_i, y_j)w_{ij}(x)
\]

\[
w_{ij}(x) = \sum_m w_m e^{-\|f_m(i) - f_m(j)\|^2}
\]

Gaussian edge potentials

\[ w_{ij}(x) = \sum_m w_m e^{-\|f_m(i) - f_m(j)\|^2} \]

• What should \( f \) be?
• simple answer: color, position
Mean field inference for Dense-CRF

\[ q_i(y_i = l) \propto \exp[-\psi_u(y_i)] - \sum_{l'} \mu(l, l') \sum_m w_m \sum_{j \neq i} e^{-\|f_m(i) - f_m(j)\|^2} q_j(y_j = l') \]

**Unary**

\[ q_i \propto \exp[-\psi_u^{(i)}] - \mu \sum_j m_{j \rightarrow i} \]

**Label compatibility transform**

**Message passing**
Fully Connected CRFs

Grid CRF

Fully connected CRF

Ground truth

sky
tree
green
Fully connected CRFs

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>

Mean field inference as a recurrent network

Figure 1. A **mean-field iteration as a CNN**. A single iteration of the mean-field algorithm can be modelled as a stack of common CNN layers.

Skip connections

Alternative: coarse-to-fine prediction

• Inspiration 1: we are making independent predictions from each layer
• Inspiration 2: CRF-like approaches require iterated inference
• Inspiration 3: Coarse-to-fine refinement works because: coarse scales capture large scale structure coarsely, fine scales capture fine-scale structures
U-Net

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Learned upsampling

- Bilinear upsampling

\[ y[i] = \begin{cases} 
  x[i/2] & \text{if } i \text{ is even} \\
  \frac{(x[(i+1)/2]+x[(i-1)/2])}{2} & \text{otherwise}
\end{cases} \]

- Assume fractional indices in \( x \) are 0
- Assume \( w[-1] = w[1] = 0.5, w[0] = 1 \)
- Then

\[ y[i] = \sum_{k=-1}^{1} w[k] x[(i - k)/2] \]
Learned upsampling

\[
y[i] = \frac{1}{\sum_{k=-1}^{1} w[k] x[(i - k)/2]}
\]

• Looks remarkably like convolution
• But output size is twice input size
• Filter w can be learnt!
• “Up-convolution”, “Transposed convolution”, “Deconvolution”
Image-to-image translation problems
Image-to-image translation problems

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

...
Revisiting contour detection
Convolutional network based edge detection

• Deep supervision: Skip connections, but additional loss at each layer

<table>
<thead>
<tr>
<th>Method</th>
<th>Max F measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured Edges</td>
<td>74.6</td>
</tr>
<tr>
<td>HED without deep supervision</td>
<td>77.1</td>
</tr>
<tr>
<td>HED with deep supervision</td>
<td>78.2</td>
</tr>
<tr>
<td>Humans</td>
<td>80</td>
</tr>
</tbody>
</table>

Convolutional network based edge detection