Fully Convolutional Networks for Semantic Segmentation

Jonathan Long*    Evan Shelhamer*    Trevor Darrell

UC Berkeley

Chaim Ginzburg for Deep Learning seminar
Semantic Segmentation

- Define a pixel-wise labeling for an image $I$ as a set of random variables $X = \{x_0, \ldots, x_n\}$ $n = \#\text{pixels}$. $x_i \in L = \text{labels} \{1, \ldots, m\}$.
- Use CNN to model a probability distribution $Q(X|\theta, I)$ over those random variables,
Problem - DCNN are great for WHAT but loose the WHERE
Naive Approach: Region-CNN

1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

“Selective Search”
SVM trained for specific class

figure: Girshick et al.
But

many seconds

R-CNN

+ Not end-to-end
FCN

< 1/5 second

end-to-end learning
History of FCN

Convolutional Locator Network
Wolf & Platt 1994

Shape Displacement Network
Matan & LeCun 1992
Predict numbers in a row

Locate the “data” square
A Classic Classification Network

Diagram of activations

227 × 227  55 × 55  27 × 27  13 × 13

convolution fully connected

"tabby cat"
Becoming Fully Convolutional

- Fully connected layers can also be viewed as convolutions with kernels that cover their entire input regions.
Becoming Fully Convolutional

- In order to get an “heatmap”, final layers need width and height so we don’t want that big kernel...
- Add a final 1X1 conv with channel for each class
Patchwise vs Whole Image

1.2 ms with AlexNet on 227X227

22ms to produce 10X10 from 500X500

100 results of classification

Convolution is fast on GPU!
Loss Function

simply the sum of chosen loss function on each pixel at the heatmap

\[ \ell(x; \theta) = \sum_{i,j} \ell'(x_{i,j}; \theta) \]
Upsampling Output

- In order to produce full image segmentation, we need to upsample the output
- Method chose: Deconvolution
Deconvolution

- Convolutional layers connect multiple input activations within a filter window to a single activation
- Deconvolutional layers associate a single input activation with multiple outputs
Upsampling

- Upsampling with factor $f$ is convolution with input stride of $1/f$
- Equivalent to backward convolution (aka Deconvolution) with output stride $f$ which is already implemented in the existing code...
- Thus upsampling is performed in-network for end-to-end learning by backpropagation from the pixelwise loss
Upsampling

- Upsample X32 in single pass by convolving with “tent kernels” - not learned!
- It has already been proven in other work that learning the kernels and upsampling gradually can achieve slightly better results
Transfer learning

“Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learnt” (Lisa Torrey and Jude Shavlik)
Transfer learning

- Cast ILSVRC classifiers into FCNs and augment them for dense prediction: discard classifier layer, transform FC to CONV, add 1X1 CONV with 21 channel dimension for score at each output location.
- Then use in-network upsampling.
- Train for segmentation by fine-tuning all layers with PASCAL VOC 2011 with a pixelwise loss.
End-to-End, Pixels-to-Pixels Network

conv, pool, nonlinearity

H × W
H/4 × W/4
H/8 × W/8
H/16 × W/16
H/32 × W/32
upsampling

H × W

pixelwise output + loss
Evaluation Metrics

Let $n_{ij}$ be the number of pixels of class $i$ predicted to belong to class $j$
let $n_{cl}$ the number of different classes
let $t_i = \sum_j n_{ij}$ be the total number of pixels of class $i$
Then we compute:

- pixel accuracy: $\frac{\sum_i n_{ii}}{\sum_i t_i}$
- mean accuracy: $(1/n_{cl}) \sum_i n_{ii} / t_i$
- mean IU: $(1/n_{cl}) \sum_i n_{ii} / \left( t_i + \sum_j n_{ji} - n_{ii} \right)$
- frequency weighted IU:
  \[
  \left( \sum_k t_k \right)^{-1} \sum_i t_i n_{ii} / \left( t_i + \sum_j n_{ji} - n_{ii} \right)
  \]
Results - Single Stream Created From Different Classifiers

<table>
<thead>
<tr>
<th></th>
<th>FCN-AlexNet</th>
<th>FCN-VGG16</th>
<th>FCN-GoogLeNet$^4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean IU</td>
<td>39.8</td>
<td><strong>56.0</strong></td>
<td>42.5</td>
</tr>
<tr>
<td>forward time</td>
<td>50 ms</td>
<td>210 ms</td>
<td>59 ms</td>
</tr>
<tr>
<td>conv. layers</td>
<td>8</td>
<td>16</td>
<td>22</td>
</tr>
<tr>
<td>parameters</td>
<td>57M</td>
<td>134M</td>
<td>6M</td>
</tr>
<tr>
<td>rf size</td>
<td>355</td>
<td>404</td>
<td>907</td>
</tr>
<tr>
<td>max stride</td>
<td>32</td>
<td>32</td>
<td>32</td>
</tr>
</tbody>
</table>
But output is coarse
Upgrade: Multi-Resolution Fusing

The scale pyramid is a classic multi-resolution representation.

Scale Pyramid, Burt & Adelson ‘83
Spectrum of Deep Features

Combine *where* (local, shallow) with *what* (global, deep)

fuse features into **deep jet**

(cf. Hariharan et al. CVPR15 “hypercolumn”)
Adding 1X1 conv classifying layer on top of pool4, then upsample X2 (init to bilinear and then learned) conv7 prediction, sum both, and upsample X16 for output.
FCN-32s, 16s, 8s

- 32s is the single stream net, the final layer is downsampled X32 \((2^5\) pooling layers)
- 16s has skip layer from pool4 (initialized with the parameters of 32s, additional params initialized to zero)
- 8s has skip from pool3
- Each net is learned end-to-end but initialize with the “coarser” nets’ params
Skip Layer Refinement

input image  stride 32  stride 16  stride 8  ground truth

no skips  1 skip  2 skips
Relative to prior state-of-the-art SDS:

- 20% relative improvement for mean IoU
- 286× faster

*Simultaneous Detection and Segmentation*  
Hariharan et al. ECCV14
Extensions

- Random fields
- Weak supervision
Fully Conv. Nets + Random Fields

- Apply CRF inference as a post-processing step

\[
E(x) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j)
\]

**Unary term**

\[
\theta_i(x_i) = -\log P_i(x_i)
\]

**Binary term**

\[
\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{K} w_m \cdot k^m(f_i, f_j)
\]

\[
\mu(x_i, x_j) = \begin{cases} 
1 & \text{if } x_i \neq x_j \\
0 & \text{otherwise}
\end{cases}
\]

\[
\kappa = w_1 \exp \left( - \frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2} \right) + w_2 \exp \left( - \frac{||p_i - p_j||^2}{2\sigma_\gamma^2} \right)
\]
Fully Conv. Nets + Random Fields

Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs.
Chen* & Papandreou* et al. ICLR 2015.
Fully Conv. Nets + Random Fields

CRF integrated into the network $f_e$

<table>
<thead>
<tr>
<th>Method</th>
<th>Without COCO</th>
<th>With COCO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain FCN-8s</td>
<td>61.3</td>
<td>68.3</td>
</tr>
<tr>
<td>FCN-8s and CRF disconnected</td>
<td>63.7</td>
<td>69.5</td>
</tr>
<tr>
<td>End-to-end training of CRF-RNN</td>
<td><strong>69.6</strong></td>
<td><strong>72.9</strong></td>
</tr>
</tbody>
</table>

[comparison credit: CRF as RNN, Zheng* & Jayasumana* et al. ICCV 2015]

**DeepLab**: Chen* & Papandreou* et al. ICLR 2015.  
**CRF-RNN**: Zheng* & Jayasumana* et al. ICCV 2015
Weak Supervision

- Sometimes we cannot use the full power of supervised deep learning, due to lack of data
- Creating semantic segmentation ground truth requires a lot of work
- However, creating “weaker” ground truth is sometimes easier to create
Fully Conv. Nets + Weak Supervision

FCNs expose a spatial loss map to guide learning: segment from tags by MIL or pixelwise constraints.

Fully Conv. Nets + Weak Supervision

- Easier to express simple constraints on the output space than to craft regularizers or ad-hoc training procedures to guide the learning.
- Such constraints can describe the existence and expected distribution of labels from image level tags (next slide).
- Use a loss function to optimize convolutional networks with arbitrary linear constraints on the structured output space of pixel labels.
Constraints on Labels Examples

suppress any label \( l \) that does not appear in the image

\[ \sum_{i=1}^{n} p_i(l) \leq 0 \quad \forall l \notin \mathcal{L}_I. \]

force some labels to appear

\[ a_l \leq \sum_{i=1}^{n} p_i(l) \quad \forall l \in \mathcal{L}_I. \]

background constraint

\[ a_0 \leq \sum_{i=1}^{n} p_i(0) \leq b_0. \]

boost all classes larger than 10% of the image by setting \( a_i = 0.1n \)
also put an upper bound constraint on the classes \( L \) that are guaranteed to be small

\[ \sum_{i=1}^{n} p_i(l) \leq b_l. \]
Fully Conv. Nets + Weak Supervision

FCNs expose a spatial loss map to guide learning:
mine boxes + feedback to refine masks.

Fully Conv. Nets + Weak Supervision

- In the training, iterate between regular training with weakly supervised data (with segmentation masks created automatically from bounding box “ground truth” labeling) and iterations where we fix the network parameters and let the suggested masks slightly change.
- It’s also possible to mix the training data with “fully supervised” data (with per pixel labeling).