Introduction to Semantic Segmentation using Convolutional Neural Networks

RADOVIC MILOS, EVERSEEN
milos.radovic@everseen.com
# Outline

<table>
<thead>
<tr>
<th>Introduction</th>
<th>Common computer vision tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Semantic segmentation - problem definition</td>
</tr>
<tr>
<td></td>
<td>Semantic segmentation vs instance segmentation</td>
</tr>
<tr>
<td></td>
<td>Applications</td>
</tr>
<tr>
<td>How does it work?</td>
<td>Representing the semantic segmentation task</td>
</tr>
<tr>
<td></td>
<td>Constructing an architecture</td>
</tr>
<tr>
<td></td>
<td>Resolving computational burden</td>
</tr>
<tr>
<td></td>
<td>Methods for upsampling</td>
</tr>
<tr>
<td></td>
<td>Transposed convolution</td>
</tr>
<tr>
<td></td>
<td>Defining a loss function</td>
</tr>
<tr>
<td>Network architectures</td>
<td>Fully Convolutional Networks for Semantic Segmentation (FCN)</td>
</tr>
<tr>
<td></td>
<td>U-net</td>
</tr>
<tr>
<td></td>
<td>The One Hundred Layers Tiramisu: Fully Convolutional DenseNets (FC-DenseNet) for Semantic Segmentation</td>
</tr>
</tbody>
</table>
Common computer vision tasks

**Classification**
In classification we predict the class of an image.

**Classification with localization**
Here we predict the class label of an image as well as position of a bounding box surrounding the object.

**Detection**
Object detection involves localization of multiple objects (that doesn’t have to belong to the same class).

**Segmentation**
Image segmentation is the process of partitioning an image into multiple segments.
The goal of semantic segmentation of an image is to label each and every pixel of an image with a corresponding class of what is being represented.

Because we're predicting for every pixel in the image, this task is commonly referred to as dense prediction.
Semantic segmentation vs instance segmentation

- Semantic segmentation does not separate instances of the same class. It only predicts the category of each pixel.

- Instance segmentation is another approach for segmentation which does distinguish between separate objects of the same class (an example would be Mask R-CNN\(^{(1)}\)).

\(^{(1)}\) Kaiming He, Georgia Gkioxari, Piotr Dollár, Mask R-CNN, CVPR 2017.
Applications

- Autonomous vehicles
- Satellite (Or Aerial) Image Processing
- Medicine
  - Organ segmentation
  - Substructure segmentation
  - Lesion segmentation
- Fashion industry, scene understanding, etc...
Representing the semantic segmentation task

- The goal of semantic segmentation is to take an image as input and output a segmentation map where each pixel contains a class label represented as an integer.

- Target can be created by one-hot encoding the class labels - essentially creating an output channel for each of the possible classes.

**Input**
- height×width×3 (RGB color image)
- height×width×1 (grayscale image)

**Target**
- height×width×N_classes

**Segmentation map**
- height×width×1
Constructing an architecture

- A naive approach: simply stack a number of convolutional layers (with same padding to preserve dimensions) and output a final segmentation map.

- This directly learns a mapping from the input image to its corresponding segmentation through the successive transformation of feature mappings; however, it's quite **computationally expensive** to preserve the full resolution throughout the network.
In order to maintain expressiveness, we typically need to reduce height and width of feature maps as we get deeper in the network. This is fine for classification but not for semantic segmentation!

Resolution: encoder/decoder structure where we downsample the spatial resolution of the input, developing lower-resolution feature mappings which are learned to be highly efficient at discriminating between classes, and the upsample the feature representations into a full-resolution segmentation map.
Methods for upsampling

- Whereas pooling operations downsample the resolution by summarizing a local area with a single value, "unpooling" operations upsample the resolution by distributing a single value into a higher resolution.
Transposed convolution

- Transposed convolution is by far the most popular approach as it allows us to develop a learned upsampling.
- In transposed convolution, we take a single value from the low-resolution feature map and multiply all of the weights in our filter by this value, projecting those weighted values into the output feature map.

Transposed convolution - 2d example

Transposed convolution - 1d example
Transposed convolution

- Transposed convolution can be represented as a convolution with some modification of the input.

Conv2DTranspose with 3x3 kernel and no padding applied to a 4x4 input to give a 6x6 output.

Conv2D with stride 1 and 2x2 padding

A Conv2DTranspose with 3x3 kernel, stride of 2x2 and no padding applied to a 2x2 input to give a 5x5 output.

Conv2D with stride 1 and 2x2 padding
Defining a loss function

- The most commonly used loss function for the task of image segmentation is a **pixel-wise cross entropy loss**.

- Problem: Model is biased towards the most prevalent class.
Soft Dice loss

- Dice coefficient is essentially a measure of overlap between two samples.
- The soft Dice loss is normalized according to the size of the target mask so it does not struggle learning from classes with lesser spatial representation in an image.

\[
\text{Dice} = \frac{2|A \cap B|}{|A| + |B|}
\]

\[
L = 1 - \text{Dice}
\]
In FCN paper\cite{Long2015} authors utilized classification networks to serve as the encoder module of the network, appending a decoder module to upsample the coarse feature maps into a full-resolution segmentation map.

Adding layers and a spatial loss (pixel-wise cross entropy loss) produces an efficient architecture for end-to-end dense learning.

FCN: From classifier to dense prediction

- **Encoder**: AlexNet, VGG, and GoogLeNet classification networks.
- **Decoder**: Transposed convolution layers.

|              | FCN-AlexNet | FCN-VGG16 | FCN-GoogLeNet
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></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>
</tbody>
</table>

Results on the validation set of PASCAL VOC 2011

Transforming fully connected layers into convolution layers enables a classification net to output a heatmap

---

FCN: Combining what and where

FCN architectures - Image from the original paper
FCN: results

Comparison of skip FCNs on a PASCAL VOC 2011 dataset.

<table>
<thead>
<tr>
<th></th>
<th>pixel acc.</th>
<th>mean acc.</th>
<th>mean IU</th>
<th>f.w. IU</th>
</tr>
</thead>
<tbody>
<tr>
<td>FCN-32s-fixed</td>
<td>83.0</td>
<td>59.7</td>
<td>45.4</td>
<td>72.0</td>
</tr>
<tr>
<td>FCN-32s</td>
<td>89.1</td>
<td>73.3</td>
<td>59.4</td>
<td>81.4</td>
</tr>
<tr>
<td>FCN-16s</td>
<td>90.0</td>
<td>75.7</td>
<td>62.4</td>
<td>83.0</td>
</tr>
<tr>
<td>FCN-8s</td>
<td>90.3</td>
<td>75.9</td>
<td>62.7</td>
<td>83.2</td>
</tr>
</tbody>
</table>

Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail.

<table>
<thead>
<tr>
<th></th>
<th>mean IU VOC2011 test</th>
<th>mean IU VOC2012 test</th>
<th>inference time</th>
</tr>
</thead>
<tbody>
<tr>
<td>R-CNN [12]</td>
<td>47.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SDS [17]</td>
<td>52.6</td>
<td>51.6</td>
<td>~ 50 s</td>
</tr>
<tr>
<td>FCN-8s</td>
<td>62.7</td>
<td>62.2</td>
<td>~ 175 ms</td>
</tr>
</tbody>
</table>

Results on the PASCAL VOC 2011 and 2012 test sets.
U-net

- Initially developed for the segmentation of biomedical images.
- The U-net\textsuperscript{[4]} architecture consists of a \textbf{contracting path} to capture context and a symmetric \textbf{expanding path} that enables precise localization.
- Won the \textit{segmenting and tracking moving cells} challenge at ISBI 2015.

\textsuperscript{[4]} Olaf Ronneberger, Philipp Fischer, and Thomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015.
U-net: loss function

- Authors used **weighted** pixel-wise cross entropy loss which force the network to learn the border pixels.

\[
p_k(x) = \exp(a_k(x)) / \sum_{k'=1}^{K} \exp(a_{k'}(x))
\]

\[
E = \sum_{x \in \Omega} w(x) \log(p_l(x)(x))
\]

\[
w(x) = w_c(x) + w_0 \cdot \exp \left( - \frac{(d_1(x) + d_2(x))^2}{2\sigma^2} \right)
\]

- \(a_k(x)\) - activation in feature channel \(k\) at the pixel position \(x\)
- \(K\) - number of classes
- \(l(x)\) - ground truth label at position \(x\)
- \(w(x)\) - total weight at the pixel position \(x\)
- \(w_c(x)\) - weight value to balance the class frequencies
- \(d_1\) - distance to the border of the nearest cell
- \(d_2\) - distance to the border of the second nearest cell

\(w_0 = 10\)
\(\sigma \approx 5\)
U-net: overlap-tile strategy for seamless segmentation

- Overlap-tile strategy allows the seamless segmentation of arbitrarily large images by splitting input images into tiles (this may help us to overcome GPU memory limitations).
- To predict the pixels in the border region of the image, the missing context is extrapolated by mirroring the input image.

Result of segmentation within the yellow lines requires image data within the blue lines as input. Solid lines refer to the first segmentation and dashed line refers to the second segmentation. By repeating this process, an entire segmentation mask will be obtained for each slice.
U-net: results

Part of an input image of the “PhC-U373” data set (left). Segmentation result (cyan mask) with manual ground truth (yellow border) - right.

Input image of the “DIC-HeLa” data set (left). Segmentation result (random colored masks) with manual ground truth (yellow border) - right.

<table>
<thead>
<tr>
<th>Name</th>
<th>PhC-U373</th>
<th>DIC-HeLa</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMCB-SG (2014)</td>
<td>0.2669</td>
<td>0.2935</td>
</tr>
<tr>
<td>KTH-SE (2014)</td>
<td>0.7953</td>
<td>0.4607</td>
</tr>
<tr>
<td>HOUS-US (2014)</td>
<td>0.5323</td>
<td>-</td>
</tr>
<tr>
<td>second-best 2015</td>
<td>0.83</td>
<td>0.46</td>
</tr>
<tr>
<td>u-net (2015)</td>
<td><strong>0.9203</strong></td>
<td><strong>0.7756</strong></td>
</tr>
</tbody>
</table>

Segmentation results (IoU) on the ISBI cell tracking challenge 2015
The One Hundred Layers Tiramisu: Fully Convolutional DenseNets (FC-DenseNet) for Semantic Segmentation

In this paper\cite{6} authors utilized ideas from the DenseNets\cite{7} to deal with the problem of semantic segmentation.

The network is composed of a **downsampling path** responsible for extracting coarse semantic features, (consisting from convolution, transition down and dense blocks) and an **upsampling path** trained to recover the input image resolution (consisting from convolution, transition up and dense blocks).

\begin{figure}
\centering
\includegraphics[width=\textwidth]{diagram.png}
\caption{Diagram of the FC-DenseNet architecture.}
\end{figure}

\begin{itemize}
  \item \cite{6} Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, Yoshua Bengio, The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation, CVPR 2017
  \item \cite{7} Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, Densely Connected Convolutional Networks, CVPR 2017
\end{itemize}
FC-DenseNet: from standard convolution to dense block

Standard ConvNet Concept

ResNet Concept

Dense Block in DenseNet with Growth Rate k Concept
FC-DenseNet: Dense block

Let \( x_l \) be the output of the \( l \)th layer.

In a standard CNN, \( x_l \) is computed by applying a non-linear transformation \( H \) (usually convolution followed by a ReLU and often dropout) to the output of the previous layer \( x_{l-1} \):

\[
x_l = H_l(x_{l-1})
\]

Pushing this idea further, DenseNets design a more sophisticated connectivity pattern that iteratively concatenates all feature outputs in a feedforward fashion. Thus, the output of the \( l \)th layer is defined as:

\[
x_l = H_l([x_{l-1}, x_{l-2}, \ldots, x_0])
\]

where \([ \ldots \) represents the concatenation operation. In this case, \( H \) is defined as batch normalization, followed by ReLU, a convolution and dropout.

Diagram of a dense block of 4 layers.
FC-DenseNet: resolving computational burden

- **Problem**: Since the upsampling path increases the feature maps spatial resolution, the linear growth in the number of features would be too memory demanding, especially for the full resolution features in the pre-softmax layer.

- **Resolution**: In order to overcome this limitation, the input of a dense block is not concatenated with its output. Thus, the transposed convolution is applied only to the feature maps obtained by *the last dense block and not to all feature maps concatenated so far*. 
FC-DenseNet: network architecture

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Transition Down (TD)</th>
<th>Transition Up (TU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input, $m = 3$</td>
<td>Batch Normalization</td>
<td>Batch Normalization</td>
</tr>
<tr>
<td>$3 \times 3$ Convolution, $m = 48$</td>
<td>ReLU</td>
<td>ReLU</td>
</tr>
<tr>
<td>DB (4 layers) + TD, $m = 112$</td>
<td>$1 \times 1$ Convolution</td>
<td>$3 \times 3$ Transposed Convolution</td>
</tr>
<tr>
<td>DB (5 layers) + TD, $m = 192$</td>
<td>Dropout $p = 0.2$</td>
<td>$stride = 2$</td>
</tr>
<tr>
<td>DB (7 layers) + TD, $m = 304$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB (10 layers) + TD, $m = 464$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB (12 layers) + TD, $m = 656$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DB (15 layers), $m = 896$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TU + DB (12 layers), $m = 1088$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TU + DB (10 layers), $m = 816$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TU + DB (7 layers), $m = 578$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TU + DB (5 layers), $m = 384$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TU + DB (4 layers), $m = 256$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$1 \times 1$ Convolution, $m = e$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Building blocks of fully convolutional DenseNets
## FC-DenseNet: results

<table>
<thead>
<tr>
<th>Model</th>
<th>Pretrained</th>
<th># parameters (M)</th>
<th>Building</th>
<th>Tree</th>
<th>Sky</th>
<th>Car</th>
<th>Sign</th>
<th>Road</th>
<th>Pedestrian</th>
<th>Fence</th>
<th>Pole</th>
<th>Sidewalk</th>
<th>Cyclist</th>
<th>Mean IoU</th>
<th>Global accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>SegNet [1]</td>
<td>✓</td>
<td>29.5</td>
<td>68.7</td>
<td>52.0</td>
<td>87.0</td>
<td>58.5</td>
<td>13.4</td>
<td>86.2</td>
<td>25.3</td>
<td>17.9</td>
<td>16.0</td>
<td>60.5</td>
<td>24.8</td>
<td>46.4</td>
<td>62.5</td>
</tr>
<tr>
<td>Bayesian SegNet [15]</td>
<td>✓</td>
<td>29.5</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>63.1</td>
<td>86.9</td>
</tr>
<tr>
<td>DeconvNet [21]</td>
<td>✓</td>
<td>252</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>48.0</td>
<td>85.9</td>
</tr>
<tr>
<td>Visin et al. [36]</td>
<td>✓</td>
<td>32.3</td>
<td>n/a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>58.8</td>
<td>88.7</td>
</tr>
<tr>
<td>FCN8 [20]</td>
<td>✓</td>
<td>134.5</td>
<td>77.8</td>
<td>71.0</td>
<td>88.7</td>
<td>76.1</td>
<td>32.7</td>
<td>91.2</td>
<td>41.7</td>
<td>24.4</td>
<td>19.9</td>
<td>72.7</td>
<td>31.0</td>
<td>57.0</td>
<td>88.0</td>
</tr>
<tr>
<td>DeepLab-LFOV [5]</td>
<td>✓</td>
<td>37.3</td>
<td>81.5</td>
<td>74.6</td>
<td>89.0</td>
<td>82.2</td>
<td>42.3</td>
<td>92.2</td>
<td>48.4</td>
<td>27.2</td>
<td>14.3</td>
<td>75.4</td>
<td>50.1</td>
<td>61.6</td>
<td>–</td>
</tr>
<tr>
<td>Dilation8 [37]</td>
<td>✓</td>
<td>140.8</td>
<td>82.6</td>
<td>76.2</td>
<td>89.0</td>
<td>84.0</td>
<td>46.9</td>
<td>92.2</td>
<td>56.3</td>
<td>35.8</td>
<td>23.4</td>
<td>75.3</td>
<td>55.5</td>
<td>65.3</td>
<td>79.0</td>
</tr>
<tr>
<td>Dilation8 + FSO [17]</td>
<td>✓</td>
<td>140.8</td>
<td>84.0</td>
<td>77.2</td>
<td>91.3</td>
<td>85.6</td>
<td>49.9</td>
<td>92.5</td>
<td>59.1</td>
<td>37.6</td>
<td>16.9</td>
<td>76.0</td>
<td>57.2</td>
<td>66.1</td>
<td>88.3</td>
</tr>
<tr>
<td>Classic Upsampling</td>
<td>X</td>
<td>20</td>
<td>73.5</td>
<td>72.2</td>
<td>92.4</td>
<td>66.2</td>
<td>26.9</td>
<td>90.0</td>
<td>37.7</td>
<td>22.7</td>
<td>30.8</td>
<td>69.6</td>
<td>25.1</td>
<td>55.2</td>
<td>86.8</td>
</tr>
<tr>
<td>FC-DenseNet56 (k=12)</td>
<td>X</td>
<td>1.5</td>
<td>77.6</td>
<td>72.0</td>
<td>92.4</td>
<td>73.2</td>
<td>31.8</td>
<td>92.8</td>
<td>37.9</td>
<td>26.2</td>
<td>32.6</td>
<td>79.9</td>
<td>31.1</td>
<td>58.9</td>
<td>88.9</td>
</tr>
<tr>
<td>FC-DenseNet67 (k=16)</td>
<td>X</td>
<td>3.5</td>
<td>80.2</td>
<td>75.4</td>
<td>93.0</td>
<td>78.2</td>
<td>40.9</td>
<td>94.7</td>
<td>58.4</td>
<td>30.7</td>
<td>38.4</td>
<td>81.9</td>
<td>52.1</td>
<td>65.8</td>
<td>90.8</td>
</tr>
<tr>
<td>FC-DenseNet103 (k=16)</td>
<td>X</td>
<td>9.4</td>
<td>83.0</td>
<td>77.3</td>
<td>93.0</td>
<td>77.3</td>
<td>43.9</td>
<td>94.5</td>
<td>59.6</td>
<td>37.1</td>
<td>37.8</td>
<td>82.2</td>
<td>50.5</td>
<td>66.9</td>
<td>91.5</td>
</tr>
</tbody>
</table>

Results on **CamVid** [8] dataset (11 classes, 367 training frames, 101 validation frames, 233 testing frames, 360x480 resolution). Architectures: (1) 56 layers (**FC-DenseNet56**), with 4 layers per dense block and a growth rate of 12; (2) 67 layers (**FC-DenseNet67**) with 5 layers per dense block and a growth rate of 16; (3) 103 layers (**FC-DenseNet103**) with a growth rate k = 16; (4) **Classic Upsampling**, an architecture using standard convolutions in the upsampling path instead of dense blocks.

### 2D models (no time)

- 2D-V2V-from scratch [34]: 55.7
- FC-DenseNet103: 79.4

### 3D models (incorporate time)

- 3D-V2V-from scratch [34]: 66.7
- 3D-V2V-pretrained [34]: 76.0

Results on **Gatech** [9] dataset (63 videos for training/validation and 38 for testing with 190 frames per video on average. There are 8 classes in the dataset.)

---

Thanks!

milos.radovic@everseen.com