Image segmentation

Shevchenko Alexander

May, 2017

Outline

1. Introduction
   - Objective
   - Purpose
   - Examples
2. FCNN
3. DeconvNets
4. MRF

Segmentation objective

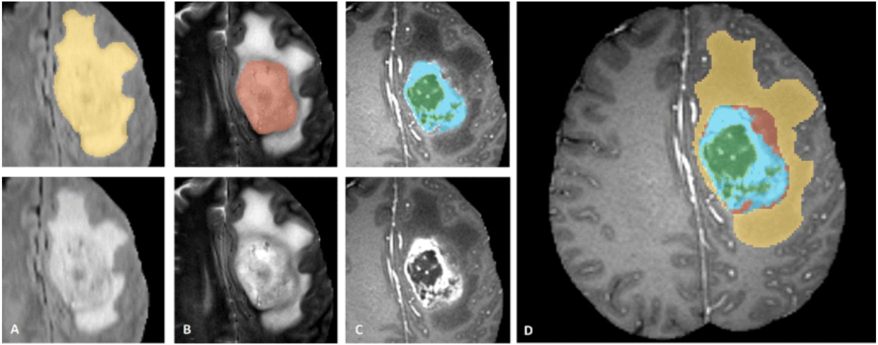
- In computer vision, image segmentation is the process of partitioning an image into multiple segments - sets of pixels, also known as super-pixels.
- More precisely: it is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

Some purposes

- Medical imaging
  - Locate tumors and other pathologies
  - Surgery planning
  - Measure tissue volumes
- Content-based image retrieval
- Object detection
  - Pedestrian detection
  - Face detections
- Recognition tasks
- Video surveillance

Examples

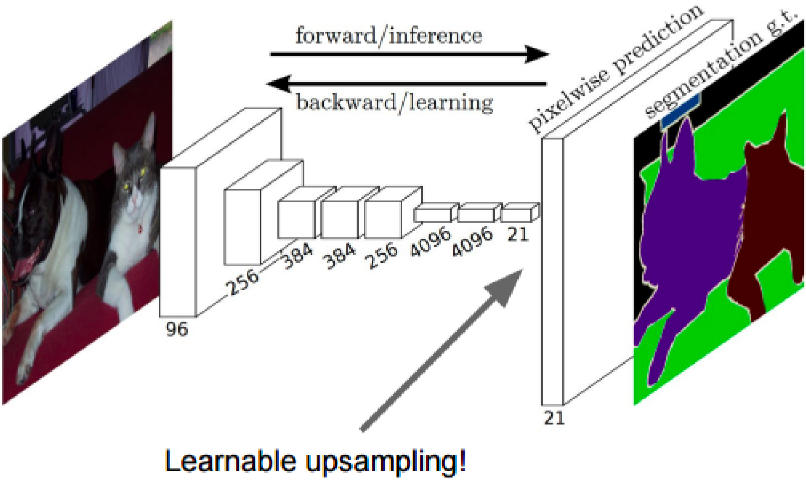
Multimodal Brain Tumor Segmentation



Examples

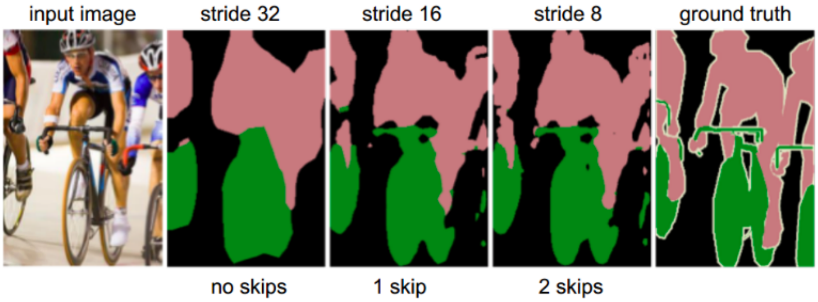
Road scene understanding

FCNN
Typical architecture



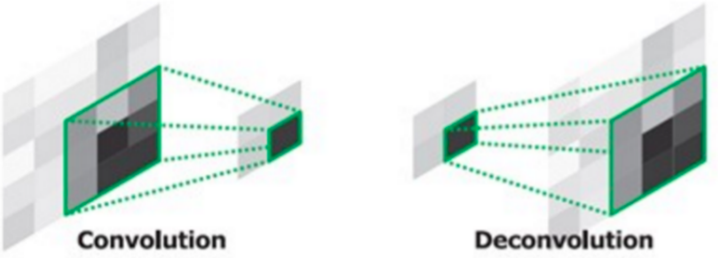
- A Fully Convolutional Neural Network is a normal CNN, where the last fully connected layer is substituted by another convolution layer with a large "receptive field". The main idea is to capture the global context of the scene
- It's important to remember that when we convert our last fully connected layer to a convolutional layer we gain some form of localization if we look at where we have more activations
- By choosing last convolutional layer to be big enough this localization effect will scale up to input image size

With simple upsampling in the end, we lose some resolution by just doing this because the activations were downscaled on a lot of steps.
Possible way to improve model is to add some skip connections:

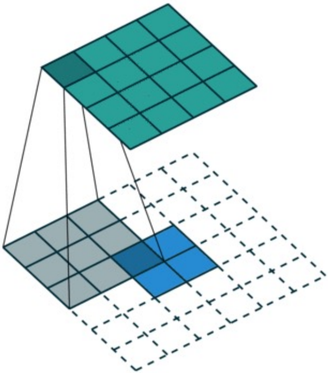


Maybe we can do better?

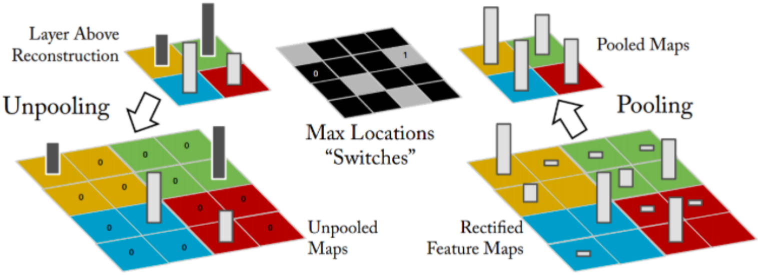
Transposed convolution



Transposed convolution



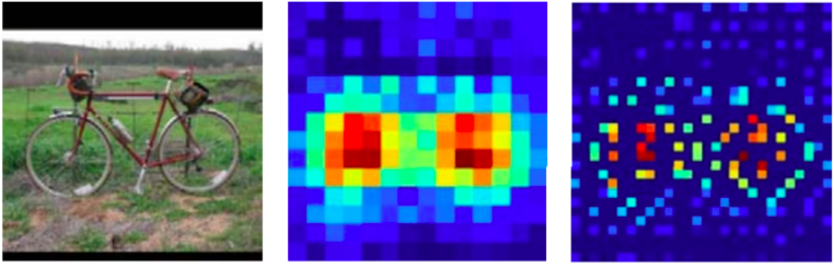
Unpooling



Layer Above Reconstruction
Unpooling
Max Locations "Switches"
Pooled Maps
Pooling
Unpooled Maps
Rectified Feature Maps

switch variables
input
pooled map
Pooling
switch variables
input
unpooled map
Unpooling

Unpooling



Deconv: 14x14 Unpool: 28x28

DeconvNets

There is another thing that we can do to avoid those "skiping" steps and also give better segmentation. Deconvnet also has better response for objects of different sizes.

Also Deconvnets suffer less than FCN when there are small objects on the scene. The deconvolution network output a probability map with the same size as the input.

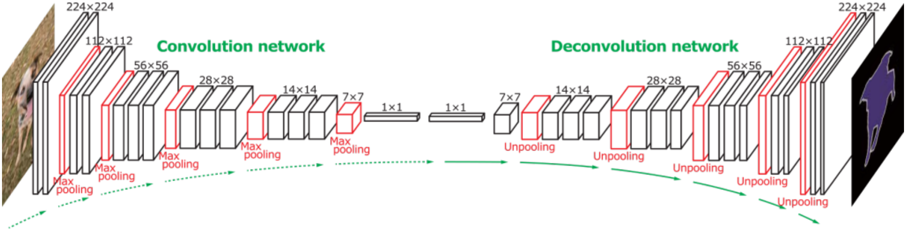


Figure: Segnet: road scenes understanding

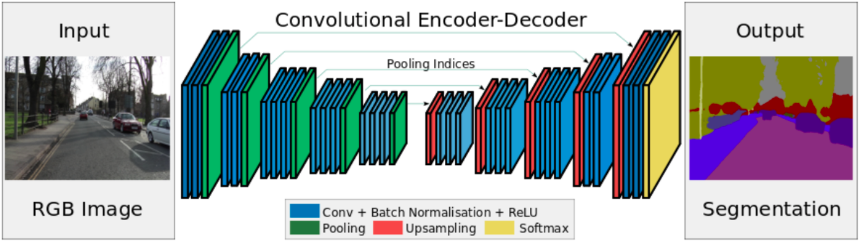
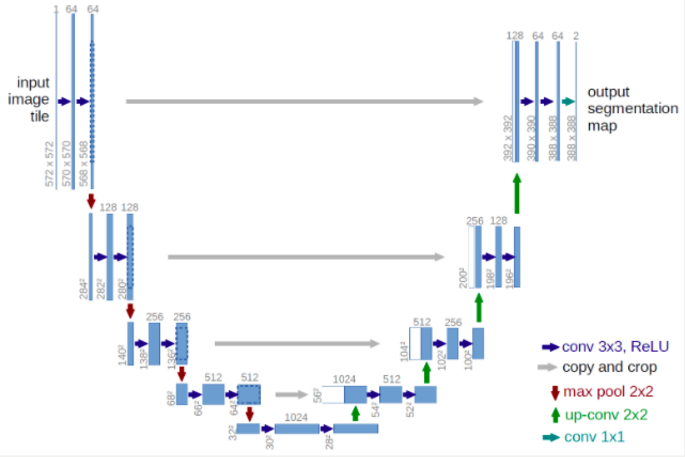


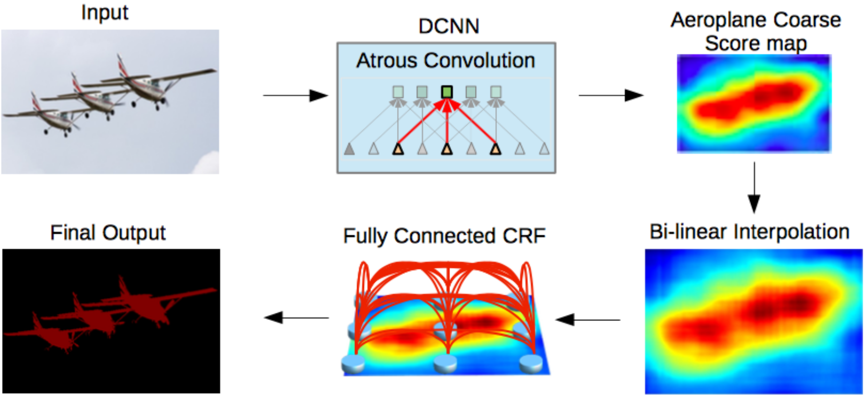
Figure: U-net: Convolutional Network for Biomedical Image Segmentation



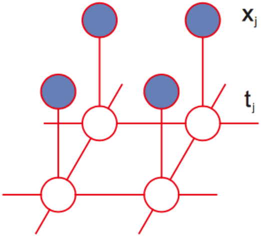
Next motivation

Improve segmentation via post-proc

Figure: DeepLab



We assume following Markov Network:



- X corresponds to observed variables (i.e. pixel intensity)
- T corresponds to latent variables, in our case label.
- We want $P(T|X) \to \max_{T}$.

Using Bayes Rule:

$$f = \arg\max_T P(T|X) = \arg\max_T P(X, T)$$

The Hammersley−Clifford theorem gives necessary and sufficient conditions under which a positive probability distribution can be represented as a Markov network.
Thus joint density can be factorized over the cliques:

$$f = \arg\max_T \prod_{(i,j)\in E} \psi_{ij}(t_i, t_j) \prod_i \psi_i(x_i, t_i)$$

.

With energy notation:

$$-\sum_{(i,j)\in E} \log \psi_{ij}(t_i, t_j) - \sum_i \log \psi_i(x_i, t_i)$$

$$= \sum_{(i,j)\in E} E_{ij}(t_i, t_j) + \sum_i E_i(x_i, t_i) \to \min_T$$

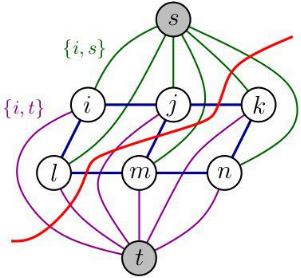
Let us assume case $t_i \in \{0, 1\}$.
Pairwise energy is submodular iff:

$$E_{ij}(0,0) + E_{ij}(1,1) \leqslant E_{ij}(0,1) + E_{ij}(1,0)$$

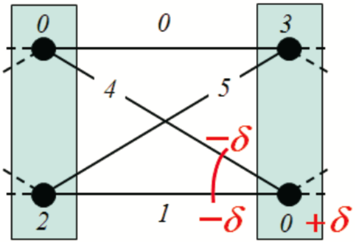
Assume flow network:

- $e(s, t_i) = E_i(x_i, 0)$
- $e(t, t_i) = E_i(x_i, 1)$
- $e(t_i, t_j) = E_{ij}(0, 1)$

Fact: In submodular case energy minimization eq. to min-cut.



Assume pairwise energy is submodular, let us make reparameterization trick:



So, we can achive:

$$E_{ij}(0,0) = E_{ij}(1,1) = 0 \text{ and } E_{ij}(0,1) = E_{ij}(1,0) \geqslant 0$$

- Potts Model: $E_{ij}(t_i, t_j) = 1 - \delta(t_i, t_j)$ - penalty for different labels
- With intensity:

$$E_{ij}(t_i, t_j) = (1 - \delta(t_i, t_j)) \cdot \exp\left(-\frac{(x_i - x_j)^2}{2\sigma^2}\right)$$

With greater intensity difference penalty is less.

Objective:

$$\sum_{(i,j)\in E} E_{ij}(t_i, t_j) + \sum_i E_i(x_i, t_i) \to \min_T$$

Latent variables is in set $\{1, .., K\}$.

- Optimizing energy in case of non-binary variables is NP-hard.
- In some special cases there is iterative procedure, which gives close to global optima solution.
- Most widely used is $\alpha-$expansion algorithm.

$\alpha-$expansion algorithm idea:

- Start with arbitrary precision
- In loop for all $\alpha \in \{1, ..., K\}$ change part of other labels to α in order to minimize energy. ($\alpha-$expansion)
- If at least for one α we reduced energy. return to step two, else break.

Necessary condition of convergence:

Theorem

Each pairwise energy term should be metric in space $\{1, ..., K\}$.

For $\alpha-$expansion step, assume similar Markov Network, but s_i corresponds to changes in value of t_i, more precisely:

$$s_j = \begin{cases} 0 & t_j^{old} = \alpha \\ 0 & t_j^{old} \neq \alpha \text{ and } t_j^{new} = t_j^{old} \\ 1 & t_j^{old} \neq \alpha \text{ and } t_j^{new} = \alpha \end{cases}$$

Thus for new network, we can construct graph and get min-cut, which minimizes energy over all possible $\alpha-$expansions.

- First of all, restrict changes in labels of class α.

$$e(S, t_i) = E_i(x_i, \alpha),\ \ e(T, t_i) = +\infty$$

$$\forall (i, j) \in E:\ \ t_i = t_j = \alpha \Rightarrow e(t_i, t_j) = E_{ij}(\alpha, \alpha) = 0$$

- For other vertexes:

$$e(S, t_i) = E_i(x_i, \alpha),\ \ e(T, t_i) = E_i(x_i, t_i^{old})$$

$$e(t_i, t_j) = E_{ij}(\alpha, t_i^{old}), e(t_j, t_i) = E_{ij}(\alpha, t_j^{old}),\ \forall (i, j) \in E$$

- Vertexes in S-cut will be assigned with α and energy will reduce.