Deep Active Learning for Interactive 3D Segmentation Of Medical Images

Vincent Groff

September 10, 2018

Table of contents

3D Image Segmentation

Medical Image Segmentation

3D Medical Image Segmentation

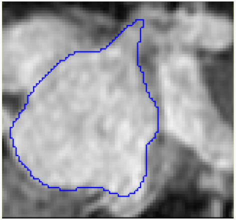

Figure 1: Segmentation of the left atrium

Main Problem: Laborious, particularly in 3D

Medical Image Segmentation

3D Medical Image Segmentation

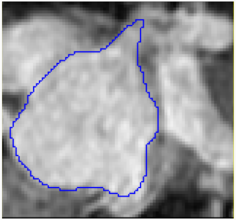

Figure 1: Segmentation of the left atrium

Main Problem: Laborious, particularly in 3D

How can we automate this?

Conditional Random Fields

Aim: Separate the set of *background* pixels ($\mathbf{B}$) from the set of *object* pixels ($\mathbf{O}$)

Define a cost function to be minimized over image X

Conditional Random Fields

$$E(\mathbf{X}) = \sum_i R_i + \lambda \sum_i \sum_j B_{ij} \tag{1}$$

- R_i - **Regional Penalty Term** - penalty for assigning pixel i to either the background ($\mathbf{B}$) or object ($\mathbf{O}$)
- B_{ij} - **Boundary Penalty Term** - penalty depending on properties of pixels i and j
- $\lambda > 0$ - a relative weighting

Conditional Random Fields

R_i - **Regional Penalty Term**

Map (0,1) probabilities to (∞, 0) penalties with negative log

Conditional Random Fields

R_i - **Regional Penalty Term**

Map (0,1) probabilities to (∞, 0) penalties with negative log

$$R_i = \begin{cases} -\log P(i \in O) & i \in \mathbf{O} \\ -\log P(i \in B) & i \in \mathbf{B} \end{cases} \tag{2}$$

Conditional Random Fields

B_{ij} - **Boundary Penalty Term**

Want to encourage boundaries at discontinuities in intensity

Penalize boundaries that are placed between similar pixels

Conditional Random Fields

B_{ij} - **Boundary Penalty Term**

Want to encourage boundaries at discontinuities in intensity

Penalize boundaries that are placed between similar pixels

$$B_{ij} = \begin{cases} 0 & y_i \neq y_j \\ \frac{1}{r_{ij}} e^{\frac{-(x_i - x_j)^2}{2\sigma^2}} & y_i \neq y_j \end{cases} \tag{3}$$

- X_i - grayscale value at pixel i
- y_i - binary label for pixel i (0 means $i \in \mathbf{B}$)
- r_{ij} - euclidean distance between pixels i and j

Conditional Random Fields

How do we solve it?

Represent the image as a graph with

- One node per pixel
- One source node (object)
- One sink (background)

Conditional Random Fields

How do we solve it?

Represent the image as a graph with

- One node per pixel
- One source node (object)
- One sink (background)

and

- Edges between pixels have a cost $\lambda \frac{1}{r_{ij}} e^{\frac{-(x_i - x_j)^2}{2\sigma^2}}$
- Edges between pixels and the source terminal cost $-\log P(i \in B)$
- Edges between pixels and the sink terminal cost $-\log P(i \in O)$

A minimum cost cut which severs the source from the sink will now minimize the cost function

Conditional Random Fields

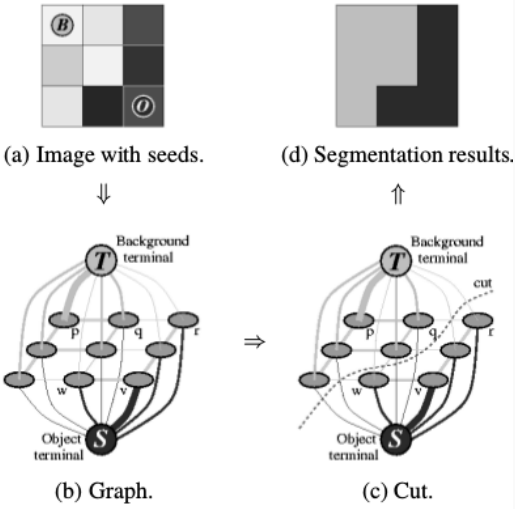

Figure 2: Minimum cost cut

Maxflow: Finding the maximal flow from source to sink

The edges that are saturated in maxflow are those in the minimal cost cut.

Conditional Random Fields

Maxflow: Finding the maximal flow from source to sink

The edges that are saturated in maxflow are those in the minimal cost cut.

Solve with **Breadth First Search**, by gradually saturating edges

Graph Cuts

How do we get $P(i \in B)$ and $P(i \in O)$?

User interaction

- User gives seed points
- Probability distribution produced from seed points
- Probabilities of other pixels inferred from distribution

Works well, but...

Graph Cuts

How do we get $P(i \in B)$ and $P(i \in O)$?

User interaction

- User gives seed points
- Probability distribution produced from seed points
- Probabilities of other pixels inferred from distribution

Works well, but...

Requires object and background pixels to have distinct intensity values

Graph Cuts

Requires object and background pixels to have distinct intensity values

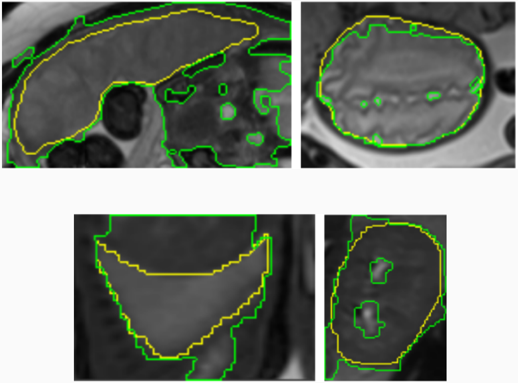

Figure 3: Grabcut algorithm in difficult cases

Graph Cuts

Requires object and background pixels to have distinct intensity values

This is the case for most traditional segmentation methods

Convolutional Neural Networks

Convolutional Neural Networks

Advantages

- Spatial operations
- Learning
- Performance

Disadvantages

- Computationally expensive
- Poor generalisation to unseen types
- Need large datasets - expensive for medical imagery

Convolutional Neural Networks

Classification CNNs - gradually extract larger features from smaller ones

Convolutional Neural Networks

Classification CNNs - gradually extract larger features from smaller ones

U-Net - gradually re-combine those features to produce a segmentation map

Segmentation CNNs

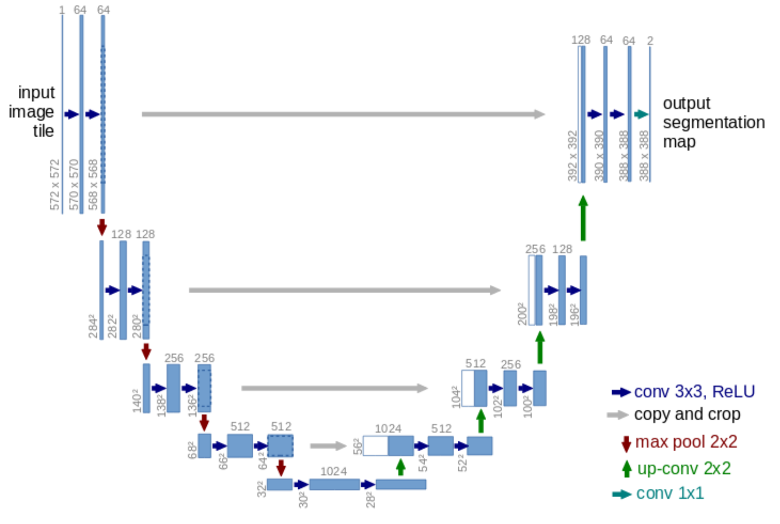

Figure 4: The U-Net architecture

Segmentation CNNs

Improvements since original U-Net:

- Residual blocks
- Spatial Dropout
- Segmentation Layers

Segmentation CNNs

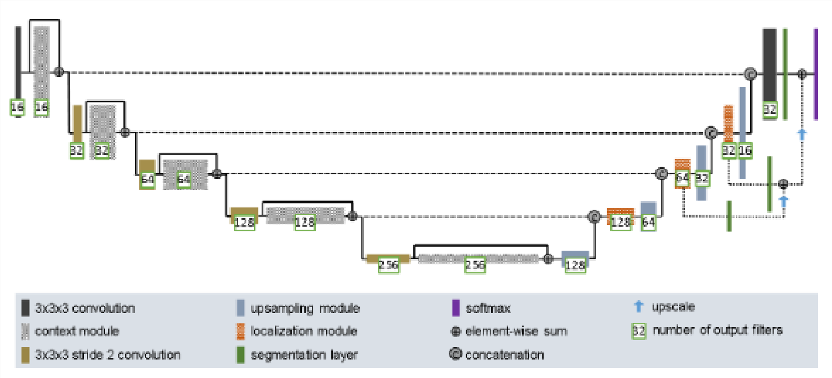

Figure 5: The Isensee et al. architecture

Segmentation Layers encourage earlier layers to produce segmentations

Aims and Design

Aims

Aims:

Minimize the disadvantages of CNNs so that they can be used for 3D image segmentation

Disadvantages

1. Computationally expensive
2. Poor generalisation to unseen types
3. Need large datasets - expensive for medical imagery

BIFSeg

Bounding-box Image-specific Fine-Tuning Segmentations (BIFSeg)
Combines CNN and CRF

Leverages user corrections

1. Draw bounding box around area of interest
2. CNN and CRF give initial guess
3. User provides correction scribbles
4. CNN fine-tunes using corrections and gives corrected result

BIFSeg

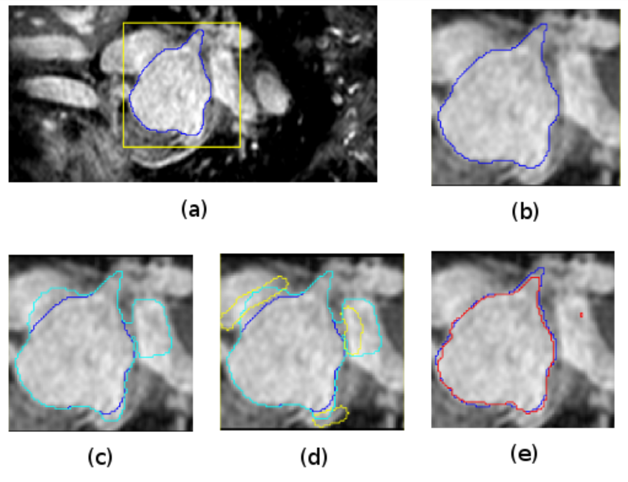

Figure 6: The steps in the BIFSeg framework. In *blue* is the ground truth, in *cyan* the initial guess, in *yellow* pixels relabelled as background and in *red* is the final result

BIFSeg

Two things going on:

- **Combining CNN and CRF**
  CNN softmax output used as probability values for calculating R_i

BIFSeg

Two things going on:

- **Combining CNN and CRF**
  CNN softmax output used as probability values for calculating R_i
- **Fine-Tuning CNN using CRF**
  1. CNN, CRF and scribbles produce segmentation
  2. CNN trained to match segmentation
  3. Repeat steps 1 and 2 n times

Fine-Tuning CNN using CRF

CNN loss function is **weighted** on a per-pixel basis during fine-tuning, such that the weight for pixel i is

$$w_i = \begin{cases} 0 & i \in \mathbf{U} \\ w & i \in \mathbf{S} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

- **U** - set of pixels for which the current segmentation is uncertain
- **S** - set of user scribbles

Fine-Tuning CNN using CRF

CNN loss function is **weighted** on a per-pixel basis during fine-tuning, such that the weight for pixel i is

$$w_i = \begin{cases} 0 & i \in \mathbf{U} \\ w & i \in \mathbf{S} \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

- **U** - set of pixels for which the current segmentation is uncertain
- **S** - set of user scribbles

Pixels in **U** are either **geodesically** near a scribble of opposite label or have P_i near 0.5

BIFSeg

BIFSeg works on unseen object types - though it needs a CNN that has learned **generalisable features**

Performs better and with less user interaction on seen object types

BIFSeg

BIFSeg works on unseen object types - though it needs a CNN that has learned **generalisable features**

Performs better and with less user interaction on seen object types

Transfer learning

1. Start by using a CNN trained in generalised segmentation
2. After *n* images have been segmented, add them to a dataset
3. Load a new CNN with the generalised CNN weights and train it on the dataset
4. Repeat steps 2 and 3 until all images have been segmented

BIFSeg

Framework:

1. Produce a small number *n* of segmentations with some generalized CNN
2. Add these to a dataset
3. Train a new CNN through transfer learning on said dataset
4. Use the new