Do not Treat Boundaries and Regions Differently: An Example on Heart Left Atrial Segmentation

Zhou Zhao  Élodie Puybureau  Nicolas Boutry  Thierry Géraud

EPITA Research and Development Laboratory (LRDE), France
zhou.zhao@lrde.epita.fr

January 15, 2021
**Introduction**

**Cardiovascular diseases** are the leading cause of death globally.
---
according to the World Health Organization (WHO).

### Medical imaging

- Computed tomography (CT)
- Magnetic resonance imaging (MRI)
- Positron emission tomography (PET)
- Single photon emission computed tomography (SPECT)
- Ultrasound (US)

MRI: **Great contrast** between soft tissues and relatively high spatial resolutions.
The difficulties of cardiac segmentation on the MR Images are shown as follow:

- **Poor contrast**
  - between myocardium and surrounding structures

- **Brightness**
  - in left ventricular/right ventricular cavities due to blood flow

- **Non-homogeneous partial volume**
  - limited CMR resolution (1.5T, 3.0T...) along the long-axis

- **Noise**
  - motion artifacts and heart dynamics

- **Shape and intensity variability**
  - different patients and pathologies
**Fully Convolutional Networks (FCN) and UNet**

- using upsampling layers and combining the feature maps from lower to higher resolutions.

---

**Main Issues**

- the lack of precision regarding the boundaries of the segmented objects.
- the loss of small objects and small parts of objects.
Loss Function: Dice and Cross-entropy (CE)

- CE: highly unbalanced problems.
- dice: difficulty dealing with very small structures.
- region-based.

There are three challenging problems applied on cardiac imaging:

- **enlarging** the receptive field of a CNN.
- **improving** the segmentation accuracy on small parts of objects.
- **balancing** the importance of the regions and the boundaries of objects.
Architecture of our networks.
**Attention Module.** $\lambda$, $\lambda'$, $\beta$ and $\beta'$ as hyperparameters, which is trained like the convolutional kernel. They decrease the weight of the unimportant feature maps.
The hybrid loss consists of two parts: region loss and boundary loss.

**Region Loss:** \( \ell_R = \ell_{CCE} + \ell_{SSIM} + \ell_{DC} \)

- **Categorical Cross Entropy (CCE)-pixel-level:**
  \[
  \ell_{CCE} = - \sum_{i=1}^{C} \sum_{a=1}^{H} \sum_{b=1}^{W} y^{i}_{(a,b)} \ln y^{*i}_{(a,b)},
  \]
  - - - multi-class classification and segmentation.

- **Structural Similarity (SSIM)-patch-level:**
  \[
  \ell_{SSIM} = 1 - \frac{((2\mu_{S}\mu_{G}+\epsilon_1)(2\sigma_{SG}+\epsilon_2))}{((\mu_{S}^2+\mu_{G}^2+\epsilon_1)(\sigma_{S}^2+\sigma_{G}^2+\epsilon_2))},
  \]
  - - - assessing image quality and is used to capture the structural information.

- **Dice Coefficient (DC)-map-level:**
  \[
  dice_i = \frac{\epsilon + 2 \sum_{n=1}^{N_i} y^{i}_{n} y^{*i}_{n}}{\epsilon + \sum_{n=1}^{N_i} (y^{i}_{n} + y^{*i}_{n})};
  \]
  \[
  \ell_{DC} = 1 - \sum_{i=1}^{C} dice_i / (N_i + \epsilon)
  \]
  - - - measuring the similarity between two sets.

The region loss guides the network to learn the transformation in a three-level hierarchy.
Boundary Loss: \[ \ell_B = \sum_i^c \int_{G_B^i} \| S_B^i (a', b') - G_B^i (a, b) \|^2 d(a, b) \]

The boundary loss function is to optimize the segmentation result.
Experiments and Results

Dataset description

- 100 annotated 3D MRIs
  - 80 cases for training and 20 cases for validating.
- Pixel spacing: $0.625 \times 0.625 \times 0.625 \text{ mm/pixel}$.
- Image sizes: $88 \times 576 \times 576$ and $88 \times 640 \times 640$ pixels.

Pre-processing

- Cropping each slice to $346 \times 346$ pixels.
- Gaussian normalization.
- Stacking 3 successive 2D frames.
Post-processing

- Keeping the greatest connected component of the segmentation.
- Padding with zeros to get back a initial width and height of a slice.

Implementation and Experimental Setup

- Keras/TensorFlow using a Nvidia Quadro P6000 GPU.
- Hybrid loss function
  - softmax to get a probability distribution over classes.
- Adam optimizer
  - $\text{batchsize}=3$, $\beta_1=0.9$, $\beta_2=0.999$, $\text{epsilon}=0.001$, $\text{lr} = 0.01$.
- Epoch=$30$.
- Evaluation method:
  - Dice coefficient;
  - 95% Hausdorff distance (95HD);
  - Average Hausdorff distance (AHD).
Results on training datasets

Comparison of our method and other state-of-the-art architectures using a 5 fold cross-validation.

<table>
<thead>
<tr>
<th>Method</th>
<th>Att. Module</th>
<th>Hyb. Loss</th>
<th>DC/%</th>
<th>95HD/mm</th>
<th>AHD/mm</th>
</tr>
</thead>
<tbody>
<tr>
<td>U-Net [3]</td>
<td></td>
<td></td>
<td>88.556(±2.586)</td>
<td>4.447(±0.996)</td>
<td>0.212(±0.077)</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td></td>
<td>89.613(±2.257)</td>
<td>4.169(±0.960)</td>
<td>0.210(±0.118)</td>
</tr>
<tr>
<td>DANet [2]</td>
<td></td>
<td></td>
<td>84.229(±3.774)</td>
<td>6.145(±2.341)</td>
<td>0.514(±0.477)</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td></td>
<td>87.584(±2.765)</td>
<td>4.903(±1.448)</td>
<td>0.280(±0.179)</td>
</tr>
<tr>
<td>Deeplabv3+ [1]</td>
<td></td>
<td></td>
<td>85.444(±3.079)</td>
<td>5.872(±2.345)</td>
<td>0.504(±0.614)</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td></td>
<td>87.556(±1.155)</td>
<td>5.210(±1.087)</td>
<td>0.273(±0.074)</td>
</tr>
<tr>
<td>Our Method</td>
<td>✔</td>
<td>✔</td>
<td>90.774(±1.568)</td>
<td>3.312(±1.277)</td>
<td>0.158(±0.092)</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td></td>
<td>91.326(±1.174)</td>
<td>3.097(±0.810)</td>
<td>0.143(±0.055)</td>
</tr>
<tr>
<td></td>
<td>✔</td>
<td>✔</td>
<td>91.792(±1.065)</td>
<td>2.868(±0.667)</td>
<td>0.130(±0.042)</td>
</tr>
</tbody>
</table>


Comparison of the proposed method and other state-of-the-art architectures. The white pixels are the differences between the prediction and the GT.
Conclusion

- **Treat boundaries and regions fairly**

- **Hybrid loss**
  - considering regions and boundaries of objects equally by combining region loss with boundary loss.

- **Attention module**
  - preventing the interferences between the surrounding similar tissues and to capture large-scale and thiner structures

- **Temporal-like method**
  - taking advantage of the temporal information by stacking 3 successive 2D frames.
Thank you!