# Prostate Segmentation on the NCI ISBI 2013 Data Set

Paul F. Jaeger

p.jaeger@dkfz.de

## 1 Project Summary

In this project, the task was to segment the central gland (CG) and the peripheral zone (PZ) on the T2-weighted MR images of the prostate provided by the NCI-ISBI-2013 challenge data set. The following section is supposed to recapture my thought process throughout the project, hence by intention lacks the formal tone of a paper:

**Process** As a first step, I analyzed the images by looking at them using MITK [1]. A remarkable observation here was the high ratio of 'empty' slices, e.g. slices that lie either above the base or below the apex of the prostate and hence only contain background class in the ground truth segmentation. Based on this observation, my first idea was to implement a curriculum training schedule, like in [2], where a sampling algorithm first focuses the training on the most informative image parts (e.g. segmented slices) and smoothly converges to the empirical data distribution. Looking at the resulting predictions on the validation set (which is a subset of the training data split by a cross-validation scheme to allow for unbiased model development w.r.t the held-out test set), this first shot 2D model showed fairly good overall performance, but one obvious weakness: The 'mid-slices' of the prostate and the distant outside slices were modelled well, but the transition between segmented and empty slices was modelled poorly, i.e. the model did not know when to 'stop' and kept drawing prostates on empty slices. Since this issue was not directly correlated to modelling the correct shape of rare class occurrences, tuning the implemented sampling schedule did not show improvements, which is why I discarded it given the short project time, so as to shift my focus to the evident problem. Therefore I looked at the input data again, observing, that even for the human eye it was not obvious in some patients, where the prostate ends, also indicating a complex task for the segmenting radiologists. When training the 2D model I had initially tried a cross entropy, a weighted cross entropy and a 2D dice loss function, but decided for the latter pretty quickly due to validation performance. The dice loss for segmentation has originally been proposed in a 3D model [3], and later been used in 2D models, but - to the best of my knowledge - there is no proposed definition on how to properly compute the dice loss over a batch of 2D slices. The technically correct way would be to compute the dice for each slice individually and average the results over the batch, which is what I did up to this point. This approach has several problems, which I was now able to correlate to the trouble the model was having when segmenting the transition slices:

- The coefficient does not consider global class occurrences, but rather class ratios in the individual slices, which introduces heavy biases.
- In empty slices the dice of the foreground classes is always zero independent of the softmax proabilities assigned by the model. As a consequence, false positives are not penalized, which causes a very poor, non-continuous gradient signal.

These issues cause the model to 'mindlessly' segment foreground classes in all slices. A possible solution is to use the batch dimension as a pseudo-volume and compute global dice coefficients over the entire batch. This modification solves both problems of the 'sliced dice loss', and additionally approximates the ultimate scoring metric (dice / IoU over the volume) more closely. Using this batch-dice-loss the model significantly ($p < 0.0001$) outperformed the classic sliced-loss (see Table 1), and produced qualitatively convincing predictions. Finally, I wondered whether true 3D information in the model could further improve the transition slices problem, so I trained a 3D U-Net architecture using the 3D dice loss computed over the volume. The resulting predictions were indeed able to model the transition slices better in some patients, but overall performance suffered from a very small batch size of 2 due to GPU memory constraints and reduction of available training samples (see Figure 1).

**Outlook** By ensembling the 2D and 3D softmax predictions, the 2D batch-dice performance could be improved by $\sim 1\%$ dice in both classes, indicating complementary value of the 3D information. In future research this information could potentially be exploited by a self-supervised auxiliary loss at the bottleneck of the 2D U-Net, which lets the model predict the relative z-position for each slice in the batch. A further promising research

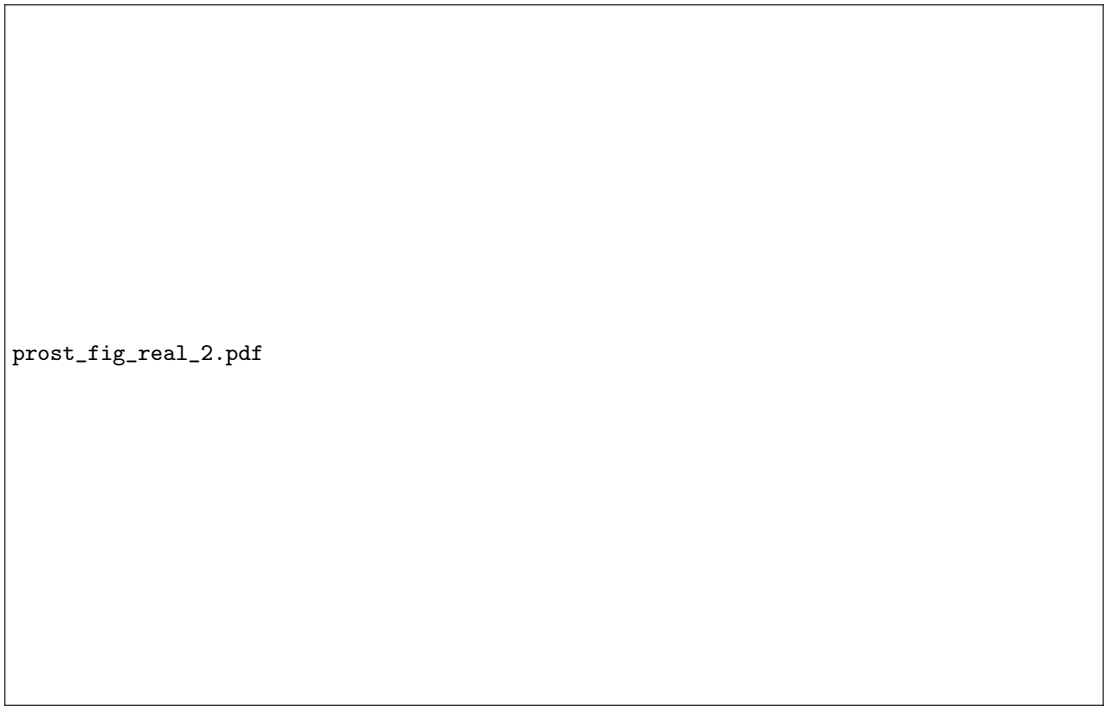


**Fig. 1.** Comparing example predictions (GC in yellow, PZ in green) of evaluated models on selected slices of two test cases. *left*: The apex of the prostate in the ground truth segmentation ends very ad-hoc (from right to left) and not obvious for the human eye. While the slice-loss 2D (dice computed for individual slices) keeps predicting foreground due to missing false-positive penalty, the batch-loss 2D (dice computed over entire batch) performs better, and the volume-loss 3D (dice over volume of a 3D model) performs best exploiting the 3D information. *right*: Example slices of a different patient show issues of the 3D model at seemingly easy structures presumably due to a reduced number of training samples.

direction could be adversarial training, as proposed by [4], which showed promising results on small and complex data sets. Since the provided data was acquired from two different scanners, one could also implement a gradient reversal layer [5], to enforce domain agnostic modelling. Of course, one could always extend the grid search of hyper parameters to e.g. a more elaborate learning rate schedule, a trade-off between model size and regularization techniques or apply modifications to the architecture like different activation functions, normalization layers, or dense connections.

**Conclusion** Despite not having found any published studies on this data set to compare my results on, based on qualitative assessment and my own experience I consider the reported results a solid baseline for further research. As a conclusion, I found this project to be a very interesting challenge with high clinical and social impact, involving numerous exciting research opportunities.

## 2 Methods

**Model** As a preprocessing step, all images were re-sampled in x and y dimensions to an equal spacing of 0.4 mm. The z-spacing was left untouched, since it deviated only marginally. One patient ('ProstateDx-01-0055') was dismissed from the training data due to a z-dimension mismatch between image and segmentation. The implemented 2D model follows the original U-Net architecture [6] except for the following deviations: leaky ReLU activation functions were used an instance normalization layer was inserted after each activation function (preferred to batch norm to avoid dangerous stochastics caused by small batch sizes). A very brief hyper parameter search identified best performance for 32 initial filters (halved the original number) and a batch size of 20. For simplicity reasons, convolutions were performed with zero-padding yielding a symmetric network with no need for feature map cropping (since no seamless tiling of output predictions was required, this was considered neglectable).

**Table 1.** Results from cross-validation (aggregated validation predictions over 5-folds) and on the test data for all methods explored in this paper. Dice coefficients and Jaccard-Index reported on the two structures of interest GC and PZ. * and ** indicate a significant difference measured via wilcoxon-signed-rank test ($p < 0.0001$). Significance is only tested on the cross-validation data, since the applied test approximates normal distribution and hence recommends $n > 20$.

| test set (n=10) | GC Jaccard (Dice) | PZ Jaccard (Dice) |
|---|---|---|
| batch-loss-2D | $\mathbf{0.780} \pm 0.061$ $(0.874 \pm 0.053)$ | $\mathbf{0.655} \pm 0.061$ $(0.790 \pm 0.045)$ |
| slice-loss-2D | $0.715 \pm 0.104$ ( $0.829 \pm 0.074$) | $0.619 \pm 0.061$ $(0.763 \pm 0.046)$ |
| volume-loss-3D | $0.739 \pm 0.089$ ( $0.846 \pm 0.064$) | $0.595 \pm 0.125$ $(0.737 \pm 0.111)$ |
| **cross.val. (n=69)** | **GC Jaccard (Dice)** | **PZ Jaccard (Dice)** |
| batch-loss-2D | $\mathbf{0.820}^{*} \pm 0.063$ $(0.900 \pm 0.04)$ | $\mathbf{0.681}^{**} \pm 0.150$ $(0.797 \pm 0.142)$ |
| slice-loss-2D | $0.757^{*} \pm 0.100$ $(0.858 \pm 0.069)$ | $0.615^{**} \pm 0.130$ $(0.752 \pm 0.117)$ |
| volume-loss-3D | $0.783 \pm 0.091$$(0.875 \pm 0.068)$ | $0.620 \pm 0.165$ $(0.750 \pm 0.162)$ |

The 3D model follows the original 3D U-Net architecture [7], except for the described deviations (with only 12 initial filters and batch size 2) and the exclusion of the first pooling in z-dimension due to limited slice numbers.

**Experimental setup** For model training the images were cropped to $288 \times 288$ pixels (and 32 slices in 3D) and heavily augmented using shifting, mirroring, rotation, elastic deformation and scaling. The model was trained in a 5-fold cross validation for 300 epochs, processing 80 batches each (20 in 3D) at a learning rate of $10^{-3}$ using the dice loss function. For inference the softmax predictions of each test case were averaged over the outputs of the 5 trained models during cross validation. Additionally, predictions were averaged over 4 different orientations of the input image obtained by mirroring. Due to time constraints, the test images were center-cropped to $288 \times 288$ pixels to be processable for the network. In this case, the assumption, that the structure of interest lies within the crop for each patient holds by a far margin, but in general all test cases should be padded to the respective next higher processable size and then cropped back to the original size after prediction.

**Software implementation** Since this was my first project in tensorflow (I am usually using pytorch), i had to get used to the symbolic graph structures first. Also it was easier for me to use my existing code framework and replace the necessary parts with tensorflow code, rather than starting from scratch or using a public library like 'tf-unet' or 'NiftyNet', especially because I tried hard to follow the 'low level'-code policy from the instructions. I did, however, use the mic-batchgenerators [8], since it is a neat, multi-threaded tool for batch generation including exhaustive data augmentation in 2D and 3D, which I helped to develop.

## References


1. I. Wolf, M. Vetter, I. Wegner, T. Böttger, M. Nolden, M. Schöbinger, M. Hastenteufel, T. Kunert, and H.-P. Meinzer, "The medical imaging interaction toolkit," *Medical image analysis*, vol. 9, no. 6, pp. 594–604, 2005.
2. A. Jesson, N. Guizard, S. H. Ghalehjegh, D. Goblot, F. Soudan, and N. Chapados, "Cased: Curriculum adaptive sampling for extreme data imbalance," in *MICCAI*. Springer, 2017, pp. 639–646.
3. F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE, 2016, pp. 565–571.
4. S. Kohl, D. Bonekamp, H.-P. Schlemmer, K. Yaqubi, M. Hohenfellner, B. Hadaschik, J.-P. Radtke, and K. Maier-Hein, "Adversarial networks for the detection of aggressive prostate cancer," *arXiv preprint arXiv:1702.08014*, 2017.
5. Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *ICML*, 2015, pp. 1180–1189.
6. O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *MICCAI*. Springer, 2015, pp. 234–241.
7. Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *MICCAI*. Springer, 2016, pp. 424–432.
8. "Mic batch generators," https://github.com/MIC-DKFZ/batchgenerators.