Medical image segmentation on Prostate MRI data

Georgia Sousouri
16-903-965
sgeorgia@student.ethz.ch

Rishabh Singh
19-953-793
risingh@student.ethz.ch

Jannik Gut
15-941-909
jgut@student.ethz.ch

I. Overview

The third homework of the course Machine Learning for Healthcare deals with a medical image segmentation problem for prostate MRI images. We were tasked to do so with a U-Net and as an additional hurdle to overcome, the different images could be rotated. In the README.md file you may find detailed description on how to navigate through the submitted notebooks.

II. Rotational invariance

We came up with two ideas to achieve rotational invariance- one using SIFT features and the other based on data augmentation.

A. Computer vision

The solution, which was also later presented in the lecture for this problem, was to use computer vision techniques. We did our experiments with ORB features, which are similar to SIFT features, but open-source and then match those features in two different images. The first set of experiments was to align the rotated test images with the help of their adjacent images. Looking at the resulting files (perfect_....), one can see that that experiment was a success and the code seems to work.
The real test is to use the training images to align the test images. Looking at the training images, one can see that they differ by quite a bit; so using something like an average image would not make sense, instead we opted to use the best match from training set as the orientation of the test image. The goodness of match is described by the accumulated difference of distances of the features. The resulting images (015_....) are very warped and nothing can be recognized from them, that is why we abolished this attempt and went through with our second idea.

B. Data augmentation with random rotation

The other idea was to create an augmented data set with deliberately randomly rotated images and accordingly rotated labels, since the rotated test set provided had also random rotations. In that way, we would achieve to make our model rotation invariant. The rotations were done without changing the image dimensions, so the labels remained centered, and the edges were padded to zero. We first chose to make 100 copies of every image (and their labels) and rotate them randomly (same angle of rotation for image and its corresponding label). We switched to 20 copies per image to have feasible training time of the net. Finally, we split the augmented dataset to training and testing sets to validate our models.
Since the values seem to deviate from image to image, we chose to normalize the values from 0 to 1 for every single image. This normalisation along with augmentation constituted our pre-processing stage.

III. U-Net

A. Net architecture

Following a tutorial, we implemented our U-Net architecture for variable depths and other hyperparameters such as filters, dropout, use of batchnorm, kernel size, activation function etc. We tested the values of critical parameters like depth and number of filters using cross-validation, while using the values of ”stock” parameters from the U-Net paper. We further modified the architecture to align with our objective of 3-class classification using *Softmax*. We used two 2D convolutional layers both with 3 dimensions of output. The output of the final convolutional layer is reshaped into vector of length equal to the multiplication of the two image dimensions (256x256). We provided some diagrams of the architecture in the folder ”/diagrams”.

B. Loss

We experimented with categorical cross entropy loss with one hot encoded ground truth masks. The results had poor performance for class 1, hence we switched to sparse categorical cross entropy loss which performed better for the dataset. Providing weights based on inverse frequency of samples in each class improved the performance further. We also implemented dice-loss, but it didn't generate results as expected.

C. Training the U-Net

We used the normalized and randomly rotated data as input to the net. We also vectorized the training labels (by 256x256).We further used a validation split of 0.2 and monitored validation accuracy. Finally, due to highly imbalanced classes, we inserted weights for each class computed based on the occurrence of pixels in each class. The classes with lesser representation in the training set were given higher weights.

IV. Validation

We observed the trend of training and validation loss with the epochs and found that learning curve flattens quite heavily (refer to 3 in appendix), three epochs also can give viable results, especially for hyper parameter tuning.
During the validation period, we also tested with weights and we found that providing weights works better than no weights or any others that we have tested.

V. Results

A. Performance Analysis

We implemented grid search for tuning hyper-parameters. We varied the depth of the network and the number of the output filters in the convolution. Due to time limitations we ran separately some sets of parameters, though, keeping fixed the dropout rate to 0.25 and the kernel size to 3. In Table I we report the overall precision (OP), per-class precision (CP) and intersection over union (IoU) scores on the augmented test data for all the iterations. For CP and IoU the unweighted averages are presented, nevertheless, in the

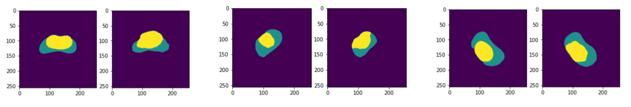
TABLE I: Evaluation metrics from grid search on test data from augmented dataset. The names in the first column indicate the depth of the U-Net (3, 5 or 7) and the number of filters (16 or 32). The w/nw indices refer to training implemented with class weights (w) or not (nw). In the parenthesis there is the last part of the name of the notebook you may find the respective iteration (first part is "Train_Evaluate_Unet_").

Depth, Filters, w/nw (file)	Overall precision	Per-class precision	IoU
3, 16, w (Iter_1_w)	0.95	0.72	0.49
3, 16, nw (Iter_1_nw)	0.95	0.69	0.49
3, 32, nw (Iter_3_nw)	0.96	0.75	0.60
5, 16, nw (Iter_2_nw)	0.93	0.75	0.50
5, 32, w (Iter_4_w)	0.97	**0.86**	0.67
5, 32, nw (Iter_4_nw)	0.96	0.57	0.50
7, 32, nw (Iter_5_nw)	**0.98**	**0.86**	**0.71**

TABLE II: Evaluation metrics from final (7,32, nw) U-Net on provided test set

	Aligned data	Randomly rotated data
Overall precision	0.95	**0.96**
Per-class precision	**0.82**	0.80
IoU	0.54	**0.57**

notebooks you can find the weighted averages as well. Validation trials gave the best results for a depth of 7 and number of filters 32 with no class weights used. As examples of how weights improve the performance, we can compare the performance for the pairwise cases with and without weights in Table 1. Adding weights led to an improved performance. In Table 2 we report the results of the best performing U-Net on both the aligned and randomly rotated test data provided for evaluation. We furthermore analysed the qualitative performance of the network, beyond just numbers, and compared the predicted mask with the ground truth labels for one instance each from augmented validation, aligned test and rotated test set as shown in Figure 1. Finally, we analysed the performance for each class in terms of the confusion matrices from the best performing U-Net on augmented, aligned and randomly rotated test sets in Figure 2.



(a) Test set (b) Rotated test set (c) Validation set

Fig. 1: Pairs of ground truth (left) and predicted (right) labels for the (a) aligned test, (b) rotated test and (c) augmented validation sets. The model has consistent performance across three sets we used for evaluation.

B. Discussion

Based on our literature survey, experiments and results obtained, we discuss some of our conclusions derived on this dataset:

- Providing labels in sparse categorical format performed better than one-hot encoded ground truth.
- U-Net is made invariant to rotated samples by augmenting training dataset with random rotations.
- Medical image segmentation is a heavily imbalanced classification problem, where the background is the dominant class.
- Providing weights to classes during training improves the performance in case of class imbalance such as this dataset.

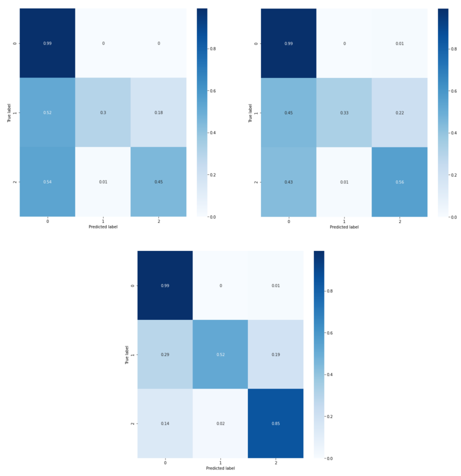


Fig. 2: Confusion matrix obtained from predictions on Aligned test (top left), Rotated test (top right) and Augmented validation set (below). The performance for classes 0 and 2 are better than class 1.

- Increasing the depth of the U-Net improved the performance for individual classes.
- Higher performance is obtained from a bigger number of filters.
- Based on the confusion matrices, class 0 is always predicted correctly. We improved upon the mis-classifications happening in rest of the two classes by instrumenting techniques mentioned above. Class 1 has the least performance and one of the plausible reasons could be the shape of the region in which pixels of class 1 occur. CNNs have a higher tendency to consider spatial locality, whereas we can see a discontinuity in the regions for class 1. The network thus has a tendency to confuse class 1 pixels for either background or class 2, especially at discontinuities, as verified by the confusion matrix. One solution could be to incorporate texture/ additional knowledge from medical domain in the ground truth for a multi-modal segmentation.

VI. Contributions

- **Georgia** had the idea and realized the augmented data set, further, she helped by finding a good multi class decision function for the U-Net, the loss function and the reshaping/vectorization of the labels.
- **Rishabh** helped in U-Net architecture, and coded custom loss functions too. He also computed the class weights and did the appendix study on different loss functions used in medical image segmentation.
- **Jannik** tried out the computer vision approach, helped by building the net part of the U-Net and ran many computations.
- Over the course of the project we all checked and validated each others code and had numerous meetings to present our individual results, discuss how to approach the upcoming problems and share the work almost equally. We appreciate the experience gained in working with git, scheduling meetings, sharing work and the knowledge we reaped from this project.

Appendix

We also evaluate the performance of U-Net on the prostate dataset provided to us using various loss functions. Loss functions play a critical role in medical datsets, where pixel distribution can be

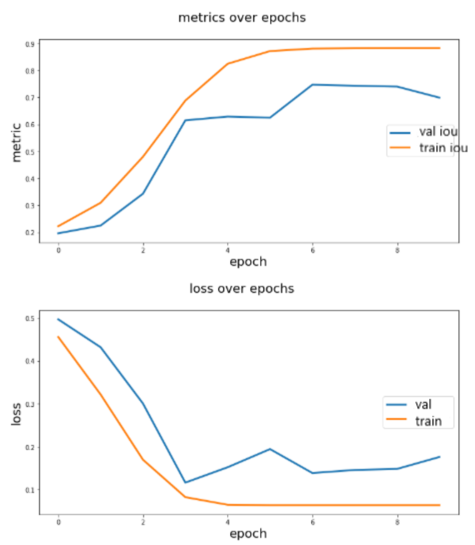


Fig. 3: The performance of final model. The IOU increased up to epoch 5 and saturated beyond, similarly the training loss almost stopped changing after epoch 5

heavily skewed. Moreover, the background class (usually annotated as '0') is often the dominant class, but is not useful for practical inferences. Thus, proper weights assigned to classes while training can force the model to focus on only relevant classes. We utilise keras' implementation of some losses and analyse their performance on a small dataset as shown in table ... These loss functions are briefly discussed below:

$$L(gt, pr) = -gt \cdot \log(pr) \tag{1}$$

$$L(gt, pr) = -gt \cdot \alpha \cdot (1 - pr)^{\gamma} \cdot \log(pr) \tag{2}$$

$$L(gt, pr) = 1 - \frac{gt \cap pr}{gt \cup pr} \tag{3}$$

$$L(precision, recall) = 1 - (1 + \beta^2)\frac{precision \cdot recall}{\beta^2 \cdot precision + recall} \tag{4}$$

where equations 1-4 refer to Categorical Cross Entropy (CCE), Categorical Focal (CF), Jaccard (J), and Dice (D) losses respectively. gt refers to ground truth and pr refers to prediction in the equations above. α, β are hyper-parameters.

TABLE III: Evaluation of different loss functions for U-Net architecture on validation set

	IOU
C-Cross Entropy	0.31
C-Focal	0.28
Jaccard	0.27
Dice	0.23

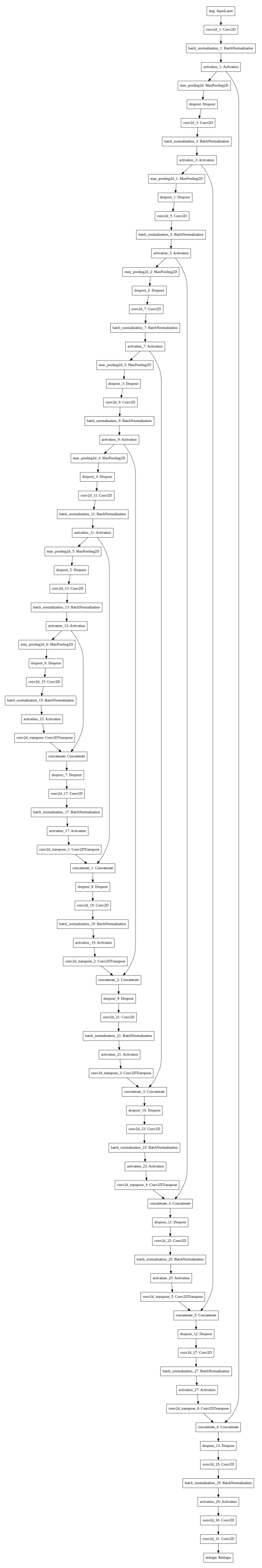


Fig. 4: U-Net architecture used in this study