ADAM

Inteneural solution description of Aneurysm Detection
And segMentation Challenge

This document is a documentation of a Inteneural team's solution for aneurysm segmentation of TOF images using Neural Networks.

Preprocessing and data representation

In the preprocessing pipeline we first reorient each nifti image to RAS and later resize the volume to match the voxel size of 0.4 in all dimensions. We load *skull-stripping** masks and exclude everything which is not our region of interest (outside of the brain). But in order to do that we need to pad each volume to match the size of the mask (it is due to the minimum size required to input the volume to the network). We use *min_max_normalization* to obtain the values between 0 and 1 (float). We perform the same operations on aneurysm and *blood vessel** masks, however, while resampling the volume to match the voxel size we rather use the nearest neighbour method. Subsequently, if the volumes we obtained are bigger than 512 (in any dimension) we cut out the center of it that matches the size. On the other hand, if volumes are smaller than 512 (in any dimension) we pad them with 0s. Finally, we save only slices which contain something (some part of aneurysm) as png files (in each orientation, axial, coronal and sagittal).
Later, in the process of training the model we use prepared slices. We have combined slices of raw TOF scan and blood vessel segmentation as input and as gold standard we have aneurysm segmentation. The shape of the input will be 512x512x2 (2 channels) and the shape of the output will be 512x512x1.

Skull stripping

For a skull-stripping task a previously trained model was used. It was pretrained on a much bigger dataset and now was fine-tuned on 13 manually labeled examples from ADAM challenge dataset. In order to further improve the continuity of the masks, a simple image processing method was used. Namely, after predicting a mask by the model external contours were detected and so obtained shape was filled with ones.

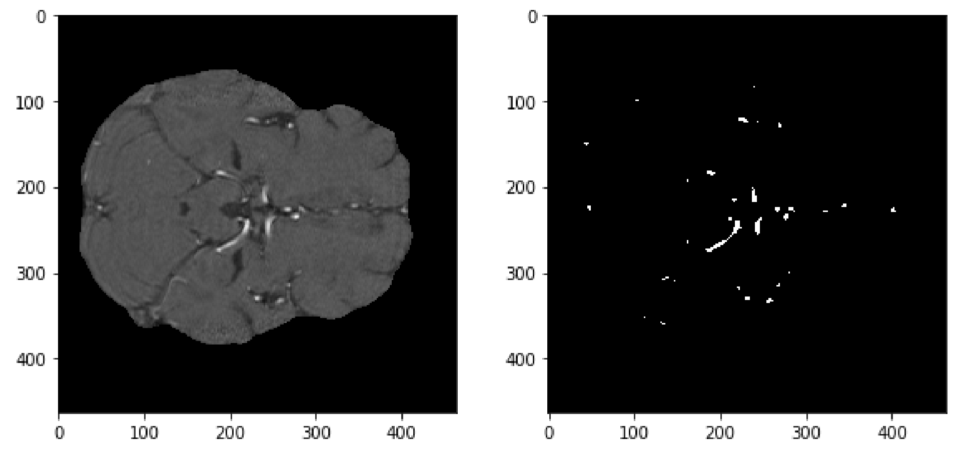
Blood vessel segmentation

In order to enrich information used by the network to learn we developed automatic blood vessel segmentation tool. In this section the idea of automatic blood vessels segmentation is described.

First, TOF images are preprocessed in the same way as described before, resampling to 0.4 voxel size, skull stripping and so forth. However, right before skull stripping, a new volume is derived by applying a threshold calculated as 99.5-th percentile of the original image and binarizing the response. Values above this threshold are now equal 1 and below are equal 0. Subsequently, after performing skull stripping on a original volume a Jerman filter is applied with parameters described below:

- *sigmas_range=(0.95, 1.3, 1.65, 2)*: scales of image
  To enhance the local structures of various sizes, the analysis is typically performed on a Gaussian scale space of the image, a novel enhancement filter based on ratio of multiscale Hessian eigenvalues, which yields a close-to-uniform response.
- *jerman_threshold=0.5*: is a cutoff threshold between zero and one. By lowering τ from one towards zero the uniformity of the filter's response rises (higher MedNR).
  At the same time, the AUC-PR decreases, however, the difference is insignificant in the range of τ = [0.5 1].

Finally, the result of applying a threshold (99.5-th percentile) and the result of applying Jerman filter (with adaptive thresholding) are combined by using simply element-wise multiplication. In addition also binary dilation operation is performed (with default parameters), a skull stripping mask is applied and filtering out small volumes is performed. By filtering out small volumes we mean keeping only the biggest 4 ones unless they are smaller than a certain threshold (equal 10237). The outcome is presented below.



Example of applying the method of automatic brain blood vessel segmentation, on the left raw TOF scan after skull stripping, on the right, the result of applying the aforementioned method. Examination 10023

NN model

We use *segmentation_models* library for creating 3 already pre-trained NN models for aneurysm segmentation (one for each orientation, axial, coronal and sagittal). Models' backbone is *efficientnetb1* which was trained on *ImageNet* benchmark and is of state-of-the-art performance in this task.

While solving the task we need to also address a class imbalance problem. In order to mitigate it we use a combination of generalized dice loss and surface loss (or boundary loss) to deal with:

$$alpha * generalized\_dice\_loss + (1 - alpha) * boundary\_loss$$

Generalized dice loss is implemented according to the article:

$$\mathrm{DL}_2 = 1 - \frac{\sum_{n=1}^{N} p_n r_n + \epsilon}{\sum_{n=1}^{N} p_n + r_n + \epsilon} - \frac{\sum_{n=1}^{N} (1-p_n)(1-r_n) + \epsilon}{\sum_{n=1}^{N} 2 - p_n - r_n + \epsilon}$$

Class weights are computed separately for each batch of images.
Boundary loss is implemented according to this article.
This loss function consists of two factors, generalized dice loss and boundary loss, multiplied respectively, by $alpha$ and $(1 - alpha)$ coefficients. Generalized dice loss is a region-based loss function. On the contrary, boundary loss takes the form of a distance metric on the space of contours (or shapes), not regions. It can mitigate the issues related to regional losses in highly unbalanced segmentation problems. The experiments showed that conjunction of these functions evaluated the best results.
Moreover, the $alpha$ coefficient was initially set to 1, and then decreased by 0.01 after each epoch, as a result the contribution of generalized dice loss was more important at the beginning, and the contribution of boundary loss was more important at the end of training.

Optimizer used for training the networks is Adam with default parameters.
For each orientation, we have a separate dataset consisting of slices with ‘something’ in them.
We split the dataset into training validation and test in the proportion: 8:1:1.
During the training we monitor loss on the validation set and keep track of the best model according to this value, in order to end up with the best model overall.

Joint prediction

Joint prediction is performed by combining 3 individual models’ predictions using simple mean. After preprocessing of a given volume according to the process described above we reorient the volume to appropriate orientation for each NN model and after performing the predictions we reorient the outputs (from axial, coronal and sagittal) to RAS again. Then we combine the predictions in the aforementioned way and perform binarization. Namely, for each voxel we check if the probability of belonging to the class 1 (aneurysm) is higher than 0.5 threshold, then a voxel is considered to include an aneurysm.

Finally, to further reduce false positive count we apply simple filtering, based on the knowledge of maximum aneurysm count in a single image (4) and minimum volume of an aneurysm (9).