Probabilistic Segmentation and Detection of Aneurysm from brain MRA with an Ensemble of 3D Convolutional Neural Networks and Monte Carlo Dropout.

Corentin Giroud[1,2] and Florian Dubost[1]

[1]Zelos Mediacorp, Rotterdam, Netherlands
[2]Ecole des Mines de Saint-Etienne, Saint-Etienne, France

August 17 2020

1 Method overview

1.1 Method.

We first created probability maps highlighting the positions of aneurysms using an ensemble of 4 networks and Monte Carlo Dropout [2]. We created a segmentation by thresholding the probability maps at 0.8. For the detection, we computed the centers of mass of the segmented connected components. The networks were 3D variants of the standard segmentation U-Net [1], trained of different random splits of the dataset. We trained 3 networks per split and selected the networks that had the best combination of sensitivity and false positive detections. It takes about 3 minutes in average to predict the aneurysm segmentation with our method, when running them on a NVIDIA 1070 GPU.

1.2 Dataset.

We used the training set of the Aneurysm Detection And segMentation Challenge, MICCAI 2020, to train our algorithms. We used the TOF-MRA images of 113 subjects without preprocessing. The image dimensions ranged from 256 to 1024 voxels in x and y and from 64 to 180 voxels in z. The voxel sizes ranged from 0.2 to 0.6 mm in x and y and from 0.4 to 1 mm in z. We only used images available in the challenge's training set.

2 Preprocessing

We resized all images to 512x512x96 voxels with linear upsampling and downsampling. Then, for each image, we extracted a patch of 224x224x56 voxels centered on the voxel at position x=247, y=208 and z=47. This was the average position of the aneurysms in the brain in the training set. Lastly, we rescaled image intensity values between 0 and 1 using 1-99th percentile normalization.

3 Convolutional Neural Network

3.1 Model Architecture.

We used a 3D variant of U-Net [1], a standard convolutional neural network for segmentation. The network consisted of a contracting path and an upsampling path. In the contracting path, there were 3 series of 2 successive convolutional layers followed by a maxpooling layer. The contracting path ended with a single convolution layer of 256 feature maps. The first pair of convolutional layers computed 8 features maps

for each convolution, the second 32, and the last 128. After the last convolution of the contracting path, we added a Batch Normalization layer [6] with Keras' default parameters [3] followed by a Dropout layer [5] at 20 percent. Then followed the upsampling path, with successively a trilinear upsampling layer, a concatenating skip connection with the feature maps of the contracting path at the corresponding resolution, and a convolutional layer until reaching the resolution of the input image. Each convolution of the upsampling path computed 8 feature maps. Each convolution in the network had a filter size of 3x3x3 and was followed by a ReLU activation. The network ended with a Batch Normalization layer [6] with Keras' default parameters [3], a Dropout layer [5] at 20 percent, a 1x1x1 convolutional layer computing a single feature map, and a sigmoid activation to rescale the value between 0 and 1.

3.2 Training.

Models were trained with a Dice similarity loss function on randomly selected train and validation splits. We used Adadelta optimizer [4] with Keras' default parameters [3]. To increase the number of learning examples, we generated random transformation of existing ones with data augmentation. During training, on-the-fly random translations, rotations and flipping were used. Training one model lasted 18 hours on a single GPU. Models were trained until convergence of the loss function evaluate on a separate validation set. The model achieving the best Dice loss function on the validation set across epochs was selected.

3.3 Prediction.

During inference, we used Monte-Carlo dropout [2]. Dropout was applied at both during training and inference. During inference, the prediction was no longer deterministic, but depended on the randomly chosen neurons. Therefore, for the same image, our model could predict different maps at every inference. For each image, we averaged the predictions of 100 Dropout iterations at 20 percent to compute the probability map during inference.

4 Ensembling

We created 4 random splits of the dataset into training of 91 images and validation sets 22 images. We trained 3 networks on each of these splits, and selected the networks that achieve the best performance on their split. The probability maps predicted by each networks were then averaged into a single probability map.

5 Post-processing

The probability maps were thresholded at 0.8. We removed all connected components larger than 15x15x15 voxels, and retained those that had a volume higher that 20 voxels. Lastly, the predicted segmentations were resized to the original input image dimensions.

6 References

[1] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science, vol 9351. Springer, Cham.

[2] Wang, S., and Manning, C. (2013, February). Fast dropout training. In international conference on machine learning (pp. 118-126).

[3] Chollet, F. others, 2015. Keras. Available at: https://github.com/fchollet/keras.

[4] Zeiler, M.D., 2012. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

[5] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), pp.1929-1958.

[6] Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.