Multiple Sclerosis Lesion Segmentation using 3D Unet

Yuxi Luo

May 2019

1 Introduction

Multiple Sclerosis (MS) is a chronic disease of central nervous system, for which a patient's status requires radiologist experts to manually delineate MS lesions from the corresponding Magnetic Resonance Imaging (MRI). There are several problems for the traditional clinician approach, despite the amount of time spent per segmentation: images from different centers often have different modalities and qualities with different MRI standards; radiologist experts often have their own discretion on potential lesion areas, which adds to inter-expert variability... Therefore, performing delineation in a large scale is almost impossible, and automatic segmentation methods are in crucial need.

A reliable model should satisfy following attributes: it should be resistant to different MRI image modalities, and should reflect inter-expert variability for the final segmentation result. In addition, often public dataset on lesion segmentation contains very few available images, so the model should extract as much features as possible from the small training dataset.

This project intends to improve results from the MICCAI 2016 challenge, using 3D Unet model to provide a more accurate outputs for MS segmentation challenge.

2 Data

This project uses the training dataset provided by the MICCAI 2016 challenge. There are 3 centers that has different scanners to reflect different MRI image qualities, which provide 4 modalities: FLAIR, DP, T1 and T2. Each center provides 5 images for training, which are delineated by 7 experts to reflect inter-expert variability of manual segmentation. In total, there are 15 images for training, where each image has 4 modalities and 7 manual segmentation. The challenge provides pre-processed data, which for each image computes a consensus segmentation from 7 manual segmentation, and standardizes each image with denoising, rigid registration, brain extraction, and bias field correction.

In particular, this project uses the pre-processed data provided by the challenge. According to the results from the challenge, FLAIR provides the most precise information of the images among all the provided modalities. Therefore, this project uses FLAIR modality for the 15 training images, and the consensus segmentations as ground-truth. A summary of the dataset is presented in Table 1.

Table 1: Details of each sequence and scanner for MS dataset

Scanner	**Matrix**	**Training cases**
GE Discovery 3T	512 * 512 * 224	5
Siemens Aera 1.5T	256 * 224 * 128	5
Philips Ingenia 3T	336 * 336 * 261	5

The dataset is separated to training and testing set with a ratio of 4 to 1. In other words, the training set contains 12 images, in which 4 of the images are randomly selected from the same scanner. The testing set contains 3 images, each of them is selected from each scanner.

3 Model

3.1 Structure

This project builds on 3D Unet structure in Keras. Originally Unet structure only deals with 2D medical images, and the loss function is a combination of binary cross entropy. The model used in this project modifies the structure, so that it could take 3D image input, and use Dice score loss as loss function to improve the quality of segmentation.

The model has 4 layers of contracting and expansive paths with 32 base filters. Each layers in contracting path has two (3, 3, 3) convolution followed by ReLu activation and (2, 2, 2) max pooling, and each layers in expansive paths has a (3, 3, 3) de-convolution for concatenated matrix with current layer and its corresponding contracted layer. At the final layer, (1, 1, 1) convolution is used to map the feature vector to 2 classes of output (lesion / non-lesion). The model should take input with arbitrary shape, since images provided by different centers varies by shape. Below is a summary of model structure with input size (64, 64, 64):

Table 2: Summary of model structure

Layer	**Name**	**Dimension of output**
0	Input	(1, 64, 64, 64)
Contract 1	Conv3D + ReLU	(32, 64, 64, 64)
Contract 1	Conv3D + ReLU	(64, 64, 64, 64)
Contract 1	Max pooling	(64, 32, 32, 32)
Contract 2	Conv3D + ReLU	(64, 32, 32, 32)
Contract 2	Conv3D + ReLU	(128, 32, 32, 32)
Contract 2	Max pooling	(128, 16, 16, 16)
Contract 3	Conv3D + ReLU	(128, 16, 16, 16)
Contract 3	Conv3D + ReLU	(256, 16, 16, 16)
Contract 3	Max pooling	(256, 8, 8, 8)
4	Conv3D + ReLU	(256, 8, 8, 8)
4	Conv3D + ReLU	(512, 8, 8, 8)
Expand 3	Deconvolution3D	(512, 16, 16, 16)
Expand 3	Concatenation	(768, 16, 16, 16)
Expand 3	Conv3D + ReLU	(256, 16, 16, 16)
Expand 2	Deconvolution3D	(256, 32, 32, 32)
Expand 2	Concatenation	(384, 32, 32, 32)
Expand 2	Conv3D + ReLU	(128, 32, 32, 32)
Expand 1	Deconvolution3D	(128, 64, 64, 64)
Expand 1	Concatenation	(192, 64, 64, 64)
Expand 1	Conv3D + ReLU	(64, 64, 64, 64)
0	Conv3D	(1, 64, 64, 64)

3.2 Patch training

The model is trained on NVIDIA Tesla P40 GPU with 64GB memory.

Initially the entire images were planned to feed into the model for training. However, the dimension for the entire image is too large for such a model. After testing around the parameters, the GPU used for this project can only afford 1 layer which does not meet our requirement.

Therefore, sliding-window approach is used instead to generate patches for training. Before actual slicing, each image and its corresponding ground-truth segmentation are zero-padded with respect to the patch size, and then sliced to generate patches for training. Then, each image is sliced to patches of size (64, 64, 64) to

include a potential region of lesion, and the patch gap is set to 16 pixel to allow some overlapping regions between two patches. During the training stage, each patch is generated randomly to feed into the model.

During the actual training, each epoch takes approximately 4 hrs.

4 Training results

4.1 Training on Dice Loss with all patches

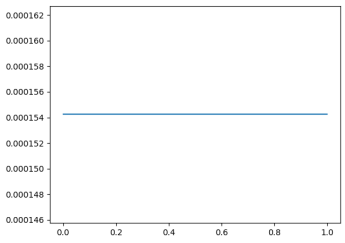
Initially the entire dataset of patches is used for training. However, the overall result after training is not promising, because most patches the model trained on do not contain any lesion region. Below are training log to train on entire dataset with dice loss:

Table 3: Training log on Dice Loss with all patches

Epoch	Training Dice Score	Training Loss	Validation Dice Score	Validation Loss
1	0.6220	0.3780	0.6039	0.3961
2	0.6385	0.3615	0.6039	0.3961
3	0.6385	0.3615	0.6039	0.3961
4	0.6385	0.3615	0.6039	0.3961
5	0.6385	0.3615	0.6039	0.3961

Even though the model is trained for several epochs, the training and validation dice score do not improve at all. The output segmentation is almost all black. Below is graph of threshold value vs. dice score, which demonstrates that the dice score, after thresholding, is still approaching to 0. Therefore, the model need to be modified so that it could learn useful information as much as possible.

Figure 1: Threshold vs Dice score



4.2 Training on Dice Loss with selected patches

Inspired by "A Multiscale Patch Based Convolutional Network for Brain Tumor Segmentation", which only uses patches that contains at least 0.01 % lesions, the data generator is modified to generate patches by that standard. The output segmentation from this approach improves a lot, and the results meet our expectation.

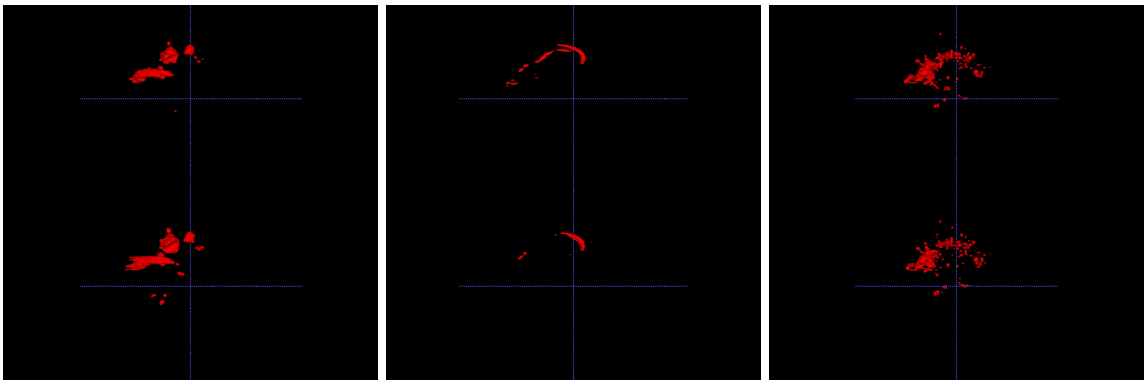
We can notice that after the fourth epoch, the model starts to overfit and the resulting dice score stays around 0.63. Therefore, weights from the third epoch are selected.

The final dice score for this model is determined after merging the pacthes back to entire image, and then calculate the overall dice with respect to ground truth segmentation. After enumerating through all possible threshold values, the optimal threshold value is 0.56, and final dice score is 0.71.

Table 4: Training log on Dice Loss with selected patches (select epoch 4)

Epoch	Training Dice Score	Training Loss	Validation Dice Score	Validation Loss
1	0.6774	0.3226	0.6014	0.3986
2	0.8791	0.1209	0.6255	0.3745
3	0.9216	0.0784	0.6227	0.3773
4	0.9409	0.0591	0.6271	0.3729
5	0.9528	0.0472	0.6272	0.3728

Figure 2: Validation images from model trained with Dice Score vs Ground truth images



(a) (128, 256, 256) (b) (192, 512, 512) (c) (320, 384, 384)

4.3 Training on Binary Cross Entropy with selected patches

The model is also modified to train on Binary Cross Entropy as the loss function instead of Dice Score, to test whether Binary Cross Entropy could serve as a better loss function for training. Since it is proved that training on all patches does not provide useful output, this model is trained on Binary Cross Entropy with selected patches.

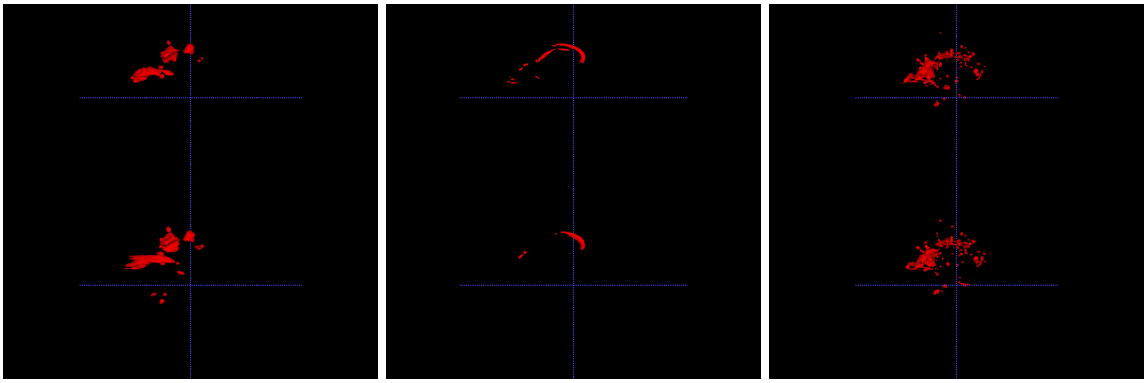
Table 5: Training log on Binary Cross Entropy (select epoch 4)

Epoch	Training Dice score	Training Loss	Validation Dice score	Validation Loss
1	0.5360	0.0273	0.5105	0.0108
2	0.8563	0.0045	0.5595	0.0127
3	0.9157	0.0028	0.5731	0.0149
4	0.9433	0.0019	0.5827	0.0174
5	0.9611	0.0014	0.5784	0.0201

The resulting validation dice score is slightly lower than the model trained with dice score directly (0.58 comparing with 0.63). However, no significant performance discrepancy is found.

Similar approach is performed to obtain the final dice score. For the ones produced by this model trained with Binary Cross Entropy, the optimal threshold value is 0.76, and the final dice score is 0.70.

Figure 3: Validation images from model trained with Binary Cross Entropy vs Ground truth images



(a) (128, 256, 256) (b) (192, 512, 512) (c) (320, 384, 384)

5 Conclusion

From the training history of the one using all patches, we learned that the model should be trained on patches that contain at least 0.01 % lesion. In addition, even though the Dice Score is used as the final evaluation metric, training on Dice Score only provides a slight advantage over training on Binary Cross Entropy. Overall, the model structure generates decent outputs even with this kind of small dataset, achieving 0.71 overall Dice Score.

In conclusion, the final dice score of Dice Loss model is 0.71 with threshold value of 0.56, and the final dice score of Binary Cross Entropy model is 0.70 with threshold value of 0.76. Even though both dice scores are relatively high, the final dice score of Dice Score model uses a more reasonable threshold value comparing to that of Binary Cross Entropy model.

It is worthy noting that the best dice score in the original challenge is 0.591. Although the testing dataset is not provided by the organizer, the resulting weight of this model should out-perform the winners of the original challenge.

6 Future works

The result of this model highly depends on the number of available training images. In the future, this project could incorporate dataset from 2015 Longitudinal MS Lesion Segmentation Challenge, which provides additional training and testing images. However, the raw data should also be processed with similar techniques used in 2016 MICCAI challenge.

There are some insights from the result of the 2016 MICCAI challenge stating that, even though FLAIR modality provides the best quality, using multiple modalities could potentially improve the result of the model. Therefore, the model could use more channels for each image to include all the modalities for training.

Since the dataset is relatively small, K-fold cross validation could be performed to select the optimal hyper-parameters for this model to further improve the results of this model.

7 Sources

- Source code on Github:
- https://github.com/Skycocoo/MS-Lesion-Segmentation